DOI 10.5162/ETTC2024/A2.1

Decentralized Reinforcement Learning for Adaptive Transmission Parameter Optimization of a LoRa Transceiver

Julius Gissing[1], *Carsten Brockmann*[2]
[1]*Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany*
[2]*Fraunhofer Insitute for Reliability and Microintegration (IZM), Gustav-Meyer-Allee 25, 13355 Berlin, Germany*
julius.gissing@izm.fraunhofer.de

Abstract:
In wireless sensor networks (WSN), a large share of the energy demand arises from wireless communication, especially in wide area networks where transmission distances are at the scale of kilometers. Ensuring reliability of communication links while optimizing energy demand requires heterogeneous radio configurations throughout the network demanding for an automated process for identifying suitable transceiver settings in order to mitigate the effort of manual configuration during deployment. Furthermore, wireless links are susceptible to dynamic influences such as environmental conditions and interference from concurrent channel usage, rendering static radio configuration impractical. Therefore, autonomous organization and self-configuration of wireless communication networks, such as transmission parameter optimization, drastically reduce cost and effort for installation and maintenance of large-scale sensor systems. Such dynamic adaptive behavior can be achieved by local execution of decentralized methods that enable decision-making at the network edge, while also inherently offering advantages such as enhanced system robustness and scalability. In this work, we present a method that exemplifies this approach and experimentally evaluate its performance on real hardware. The adaptive algorithm optimizes the transmitter configuration of a *LoRa* transceiver by employing a model-free reinforcement learning approach based on an actor-critic setup using a parameterized stochastic policy and state-value function approximation. Experimental results show that the approach surpasses a standard approach in terms of long-term energy demand. Furthermore, the method's capability of adapting to dynamic wireless channels is demonstrated.

Key words: Reinforcement Learning, Decentralized Systems, Wireless Sensor Networks, LoRa, Energy Efficiency

Introduction

The number of interconnected devices in the Internet of Things (IoT), among which, wireless sensor nodes constitute a large share, is growing steadily during the last years. In light of 25 Billion devices forecasted by the year 2030 [1], the effort for installation, maintenance and administration is tremendous. To address this challenge, devices should be highly resource-efficient, ensuring a long lifespan and extended maintenance cycles. Furthermore, autonomous configuration and organization of distributed devices improve the ease of use, save efforts during deployment and improve system robustness. Achieving these advancements requires individual devices to make decisions autonomously at runtime and in coordination with other network participants. These decisions encompass various tasks of network organization, such as establishing connectivity, selecting appropriate transceiver configurations and realizing efficient routing. This approach relies on the concept of decentralized systems, in which, unlike in centralized architectures, the edge nodes of the network contribute the essential functionality of the entire network.

This work will present an approach of reinforcement learning to support local decision making of distributed devices in decentralized systems. The paper is structured as follows. First, related work with focus on machine learning in wireless sensor networks and in specific in *LoRa* and *LoRaWAN* networks will be discussed. Next, the essential fundamentals of smart sensor systems, decentralized systems, reinforcement learning, and the networking standard *LoRa/LoRaWAN* will be outlined. Thereafter, the developed algorithm for transmission parameter optimization will be introduced and the results of experiments are

presented. Finally, the findings are summarized in a conclusion.

Related Work

Cui et al. emphasize the importance of learning and intelligent algorithms from the field of machine learning for the Internet of Things in their article [2]. The applications they discuss include device identification within networks, data security measures, network traffic prediction, and various examples from edge computing. The significance of edge computing is particularly highlighted for reducing server load, minimizing energy consumption, and supporting low-latency applications.

In [3], the authors describe a learning method for wake-up control of sensor nodes in a sensor network for compressed data acquisition. The goal is to evenly distribute energy consumption and thereby extend the network's lifespan. The approach is based on reinforcement learning, intended to be executed locally on the sensor node. The achieved results demonstrate an improvement in network lifespan compared to similar approaches. The simulation used, consisting of 512 uniformly distributed sensor nodes, assumes that all nodes know the shortest transmission path to the base station and can be woken up at any time by a radio message.

In their article [4], Zhang et al. describe the optimization of *LoRa* radio parameters aimed at achieving a fair allocation of network resources and minimizing overall network energy consumption. Their approach is based on integer linear programming, which is approximated using a combination of an evolutionary algorithm and the Gurobi solver. Compared to conventional resource allocation strategies, their simulations show a reduction in energy consumption by more than 20% while also decreasing packet collisions. The approach relies on a high-performance server and a centralized system architecture.

Farhad and Pyun [5] provide an overview of current machine learning techniques for enhanced resource management in *LoRa* and *LoRaWAN* networks. The methods described rely on learning algorithms executed by the network server within a centralized network architecture. These algorithms can therefore access information from all nodes in the network or from an offline dataset. Evaluation of the techniques is based on simulations.

González et al. [6] employ various machine learning algorithms to predict path loss and shadowing effects in a *LoRa* connection based on environmental factors. They propose a method for calculating efficient radio parameters by executing a pre-trained model on the network node. The algorithm's evaluation is conducted through simulation.

Azizi et al. introduce an algorithm for parameter optimization of a *LoRa* radio module, based on reinforcement learning and algorithms from the field of multi-armed bandits [7]. The primary goal is to optimize the packet delivery rate of message transmissions, demonstrated in a simulation involving 100 nodes. The authors limit the radio configuration to a maximum of 18 parameter combinations and demonstrate that their approach significantly improves the delivery rate compared to the standard used in *LoRaWAN*.

There are several promising approaches to resource allocation and configuration optimization in *LoRa* networks, with some occasionally adopting the approach of distributed, decentralized execution. To our knowledge, there are no existing works that implement these concepts on real hardware and evaluate them in experiments outside of simulations.

Smart Sensor Systems

With developments towards the internet of things, the notion of smart sensors included more advanced signal processing and data fusion techniques on the device, especially intelligent algorithms and concepts of artificial intelligence that improve the usability of the system in terms of integration and extraction of information [8]. In recent years the number of operational systems is quickly growing and each system may consist of a large number of end nodes, making it impossible to configure and manage each device manually. Additionally, node failure, replacement or network extensions are common practice in WSNs. Especially large-scale networks are prone to failure of single nodes due to battery depletion or electronic failures over time. Therefore, self-organizing, self-configuration and self-healing capabilities are key factors for a successful WSN deployment [9, 10]. These requirements demand the behavior of individual nodes to be more autonomous and self-adaptive, pushing sensor nodes more towards actual intelligent behavior in order to form a system that is capable of adapting to many circumstances and possibly generalizing over different application domains. Promoting intelligent behavior of individual nodes inside a WSN may enable the system to fulfill described expectations of modern sensor systems.

Decentralized Systems and Decision Making

Many techniques that are used in recent time rely on centralized cloud servers, connected to base stations that organize the network and roll

out configurations to network entities. Though the centralized approach has the advantage of using powerful servers as the network backbone and opening up the opportunity of using computationally heavy algorithms on the sensor data, it also comes with a range of issues. Each central base station has a limited number of devices that it is able to manage due to the limited bandwidth and duty-cycle. Configuration of nodes that are far away may require communication over relay nodes which further increases communication overhead. Generally, centralized control of end-devices by a base station impedes scalability of the network and decreases operational robustness. In order to address these challenges, decentralized data processing and network management has become a research focus. Decentralized network management aims to organize network nodes and resources led by end-devices, with minimal need for manual intervention. Without centralized control, individual nodes aim at optimizing global objectives by the means of localized decision making. Each node must observe its environment by using sensors and communicate with its neighbors. Based on that information and processing capabilities of end nodes the local decision making process is employed with the goal of optimal behavior independent of the place of installation and circumstances present. Sensor nodes are enabled to adapt according to the network's needs in changing conditions. Meeting this objective requires increased functionality of individual end nodes, specifically, higher complexity of firmware components that controls task sequencing and decision making. This goes along with higher demand on the computational power and resources [11].

Reinforcement Learning

Reinforcement learning (RL) is a subclass of machine learning that aims at finding a (nearly) optimal behavioral policy of an agent acting in an unknown environment. The agent optimizes its policy by the means of trial and error, supported by a reward signal that evaluates the taken actions. The goal of a learned policy is to maximize the long-term cumulative reward. The formalization of reinforcement learning is based on a Markov decision process (MDP) which is a tuple $(\mathcal{S}, \mathcal{A}, P, R)$ where the state space $\mathcal{S}$ is the set of all states in the environment and the action space $\mathcal{A}$ is the set of actions the agent can take. $P(s'|s,a)$ is the state transition probability and $R(s,s',a)$ is the reward the agent collects for transitioning from state $\boldsymbol{s} \in \boldsymbol{\mathcal{S}}$ to $\boldsymbol{s'} \in \boldsymbol{\mathcal{S}}$ given that action $a \in \boldsymbol{\mathcal{A}}$ was performed. For most environments, especially in the real world, the transition probability function is unknown, which means the agent has to explore the environment in order to find a policy that yields maximum cumulative reward. On the other hand, if an agent always explores its environment it can never exploit the behavior that is believed to generate most reward. Balancing the trade-off between exploring the environment and exploiting what has been learned to effectively gain reward is one of the fundamental problems in RL and has been studied intensively. At each time step t the agent has to choose an action a_t, given the current state $s_t \in \boldsymbol{\mathcal{S}}$, which may be exploratory or greedy. It does that by evaluating the current behavioral policy π_t that maps the current state $s_t \in \boldsymbol{\mathcal{S}}$ to an action $a_t \in \boldsymbol{\mathcal{A}}$. In general, the policy can be deterministic or stochastic and is given by a probability distribution.

$$\pi(a|s) = \mathbb{P}(a_t = a|s_t = s)$$

After taking an action the agent observes the successor state s_{t+1} and the gained reward r_t. During exploration of the environment the agent has to acquire knowledge about the potential of long-term reward of each state, in order to adapt its policy. A common concept of evaluating a state is to estimate the value function. The state-value function $V_\pi(s)$ is defined as the expectation of the discounted long-term reward under policy π when starting in state s.

$$V_\pi(s) = \mathbb{E}_{a\sim\pi, s\sim P(s'|s,a)}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s\right]$$

Note, that the expectation also takes into account the transition probability of the environment. However, this will be the case for every expectation used from here on, so the subscript will be omitted to simplify the notation ($\mathbb{E}_{a\sim\pi, s\sim P(s'|s,a)}[\cdot]$ from here on denoted as $\mathbb{E}_\pi[\cdot]$).The discount rate γ is a parameter $0 \leq \gamma \leq 1$ that determines the importance of future reward in contrast to immediate reward, besides, it is particularly important for an infinite planning horizon, because it ensures the sum to be finite. The sum of the discounted future rewards is also referred to as the return G. Similarly to the state-value function, the action-value function $Q_\pi(s,a)$ can be defined as the expected return when starting in state s, taking action a and following the policy π thereafter.

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a\right]$$

A distinctive characteristic of RL is the ability of agents to learn from observations and locally available information. As opposed to supervised

learning, there is no need for data that is labeled by an expert. An RL agent collects information and incorporates it into the future decision process, which makes it an inherently adaptive class of algorithms that is able to cope with uncertain environments. A key idea that allows to immediately learn from examples that are generated from interactions with the environment is called temporal difference (TD). TD learning takes experienced samples and uses them to incrementally approximate the state or state-value function. In order to update estimates of the value function of a certain state, they rely on their current value estimate of the next state (bootstrapping).[12]

$$V(s_t) \leftarrow V(s_t) + \eta(r_t + \gamma V(s_{t+1}) - V(s_t))$$

Where η is the learning rate (or step size) and $V(s)$ is the current estimate of the value if in state s. For a sufficiently small step size and following a certain policy during interaction, the approximation is guaranteed to converge to the actual value under that policy. When TD prediction is applied to estimate action-values instead of state-values, it can be used for optimizing an agent's policy. During operation, the action with the highest action-value ($\mathrm{argmax}_a\, Q(s,a)$) is executed. This means no more exploration of the state space and requires sufficient prior experience in the environment by the agent. Furthermore, this kind of policy leads to deterministic decisions which may lead to optimal behavior in deterministic environments. In stochastic and dynamic environments it has been shown that stochastic policies may yield to better results [13].

Instead of defining behavioral policies directly through action-values, an additional stochastic function can be defined to map states on action probabilities. Policy gradient methods define a parameterized policy and optimize the parameters with respect to the expected return the policy is generating. For a parameterized policy $\pi_\theta(a|s)$ it can be shown that an optimization step for the function weights θ in direction of the gradient of the expected return,

$$\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta V(s_t),$$

can be computed using the return and the gradient of the policy, without explicit knowledge of the transition probabilities of the environment.

$$\nabla_\theta V(s) = \mathbb{E}_\pi\, [\nabla_\theta \log \pi_\theta(a|s)\, G]$$

When the expectation is replaced by sampled data, this is called the REINFORCE estimator [14]. A characteristic of the REINFORCE estimator is its need for the full return of a state. This can become an issue for tasks that do not follow an episodic structure, since it means long trajectories need to be sampled in order to compute the policy gradient and optimize the parameters. Fortunately, instead of using the actual return, it can be replaced by an approximate of the state-value or action-value function, as it is learned in the TD-learning setup. Algorithms that use a value function estimate to update the policy are called actor-critic methods, since they learn a policy (actor) by optimizing it in direction of the value function that evaluates its actions (critic) [15]. A typical choice for the critic is the advantage function $A_\pi(s,a)$, which can be described as the advantage of taking action a over taking another action $a^* \neq a$, when in state s. It is defined as the difference between the action-value function $Q_\pi(s,a)$ and the state-value function $V_\pi(s)$.

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$$

Instead of learning a state- and action-value estimate in parallel, the advantage can be estimated by the action-value and the gained reward only.

$$Q_\pi(s_t,a_t) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \middle| s_t, a_t\right]$$

$$= r_t + \gamma \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \middle| s_{t+1}\right] = r_t + \gamma V_\pi(s_{t+1})$$

$$A_\pi(s_t,a_t) = r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t)$$

Therefore, the critic can be estimated by learning an approximate state-value function $\hat{V}_\pi(s) \approx V_\pi(s)$, for example with on-policy TD-learning. The policy optimization rule for an actor-critic algorithm is given by

$$\theta_{t+1} \leftarrow \theta_t + \eta \nabla_\theta \log \pi(a_t|s_t)\, \hat{A}_\pi(s_t,a_t).$$

Since it may take many exploration and learning steps until the state-value estimator converges to reflect the true state-value under the current policy, the state-value and policy parameters are optimized concurrently. The true value function is changing under the constantly optimized policy, as a consequence, the state-value approximation may be inaccurate. Other reasons for wrong value estimates may be delayed reward or non-Markovian property of problems [12, 16]. A standard procedure to counter these problems are eligibility traces. Eligibility traces are a discounted accumulation of the gradients that are used in the optimization step. In case of eligibility traces for the actor optimization, this means, the gradient $\nabla_\theta \log \pi_\theta(a_t|s_t)$ at times step t is not only

influencing action a_t at time step t but also at the following time steps $t+1, t+2$, ..., etc.

Aforementioned RL techniques necessitate the use of function approximation to model state value and policy function. The implementation can range from simple approaches, such as tabular methods, to complex, nonlinear representations, such as artificial neural networks. Since tabular representations require substantial memory in multi-dimensional state spaces and artificial neural networks demand significant computational power for optimization algorithms, both methods are not well-suited for resource-constrained microcontrollers. An efficient approach to function approximation is provided by linear methods in combination with nonlinear projections of the state variables [17]. One effective projection method is tile coding, which divides the state space into partitions (tiles) of arbitrary size. The partition containing the input vector is activated (1), while others remain inactive (0). Multiple overlapping tilings enable function approximation to generalize across nearby input values, as different partitions can be activated in different tilings. [18]

LoRa & LoRaWAN

LoRa (Long Range) is a proprietary modulation technology for wireless communication, developed by *Semtech that* is based on the technique of chirp spread spectrum. In this process, a communication signal is spread in the frequency domain, resulting in a signal with increased bandwidth but the same total power. Due to the enlarged bandwidth, the signal becomes less susceptible to disturbances such as multipath fading, interference, or the Doppler Effect [19]. Demodulating the signal back to the base frequency allows the receiver to benefit from processing gain, enabling the detection of signals below the noise floor with high accuracy [20]. Thus, robust communication over long distances can be achieved with minimal energy use. *LoRa* offers flexible parameterization of the radio module which enables balancing the trade-off between link quality and energy demand. Transmit power (TP), spreading factor (SF), bandwidth (BW), and coding rate (CR) can be configured to meet the requirements of specific applications. The flexible selection of parameter combinations creates a complex search problem for the optimal configuration of the radio module [21].

The recommended networking layer to establish a working communication network with *LoRa* is the open-source protocol *LoRaWAN*. In *LoRaWAN,* network participants are organized in a star topology which allows for communication exclusively between end nodes and gateways. Since there is no widespread networking protocol based on a meshed or multi-hop topology, the *LoRaWAN* standard will be the benchmark for further evaluation during this work. The standard describes a procedure for parameter optimization of the radio module, called adaptive data rate (ADR) [22]. The concept is based on the range versus data rate tradeoff and consists of increasing data rate as much as possible while maintaining a stable communication link. Higher data rate goes along with shorter transmission time, leading to less energy consumption and shorter communication range. Along with decreased energy usage, channel efficiency benefits from short time on air, resulting in larger channel capacity and mitigating risk of message collision. ADR is a link-based approach, meaning that the transmission parameters for communication between end-node and gateway are determined by the network server in a centralized fashion. If an end-device uses ADR or not is decided by the application and the process is initiated by the device. The network server collects some messages of the node in question and issues a downlink including the ADR command through the gateway with the best communication link to the node. The optimal data rate is computed by the server, taking link budget and a margin for error into account. After a predefined number of messages the node requests an ADR acknowledge from a gateway to ensure a stable connection. If the connection was lost, it gradually decreases its data rate in a standardized back-off scheme until the communication link is reestablished. The link-based ADR approach has several shortcomings that have been reported especially with increasing network size [22, 23]. ADR commands and ADR acknowledgement creates a traffic overhead in the network that leads to increased channel occupation. This lowers the packet delivery ratio due to message collisions. Lost ADR acknowledgment also leads to undesired data rate increase even if low data rates are applicable. Moreover, gateways servicing many end-devices easily exceed duty-cycle limitations on unlicensed frequency bands due to the requirement to send acknowledgment and ADR frames to all connected devices. These issues with centrally managed ADR schemes lead to interest in network-aware concepts that follow a distributed approach, meaning that end-devices determine optimal transmission parameters locally [23]. Furthermore, ADR is only applicable to static end-nodes because a constant propagation environment is assumed during link probing. If the wireless channel is dynamic, by the time an ADR command is issued through the gateway, the estimated link budget is not valid anymore [22].

Online Reinforcement Learning for Transmission Parameter Optimization in LoRa

At the time of deployment, the quality of the channels between participants of a WSN is unknown and may be heterogeneous across the network. Optimal parameter settings of a radio transceiver minimize energy consumption while ensuring communication reliability. The choice of transmission parameters can be optimized in a decentralized fashion by employing reinforcement learning (RL) on the end nodes. The exact method and procedure for this is described in the following.

First, the problem needs to be described in form of a Markov decision process (MDP) in order to make use of an RL approach. The state of the process should provide precise contextual information of the learning agent's situation. It is composed of the radio configuration of the transceiver and the measured signal quality using that configuration. The signal quality is described by the received signal strength indication (RSSI) value and the signal-to-noise ratio (SNR), it is sent as a feedback by the receiving entity. In order to move around in the state space, the agent is endowed with a set of actions that change the radio configuration in a differential way. The reward is constructed to reflect importance of successful transmissions and minimization of energy consumption jointly. In particular, a state s_t at a discrete time step t is defined as the tuple

$$s_t = (SF_t, TP_t, RSSI_t, SNR_t),$$

with $RSSI_t$ and SNR_t denoting the observed signal values when using spreading factor $SF_t \in \{SF5, SF6, \dots, SF12\}$ and transmission power $TP_t \in \{0\ dBm, 1\ dBm, \dots, 14dBm\}$ for a transmission. The agent then evaluates its policy $\pi_t(a_t|s_t)$ and chooses a differential action a_t from the discrete set of actions

$$\mathcal{A} = \{\mathrm{inc}(SF), \mathrm{inc}(TP), \mathrm{stay}, \mathrm{dec}(SF), \mathrm{dec}(TP)\},$$

where $\mathrm{inc}(\cdot)$ and $\mathrm{dec}(\cdot)$ denote increasing and decreasing of a radio parameter respectively, and stay means to remain in the current transceiver configuration. After execution of the chosen action a_t the agent observes the successor state s_{t+1} and the gained reward r_t that is given by the following function.

$$r_t = \begin{cases} E(SF_t, TP_t)^{-1}, & \text{if transmission successful} \\ -p, & \text{if transmission not successful} \end{cases}$$

$E(SF, TP)$ provides the energy required for a transmission with spreading factor SF and transmission power TP. Therefore, the reward is positive for all successful transmissions, inversely proportional to the required energy and constant negative for unsuccessful communication attempts.

The first part of the algorithm is to estimate the state-value for any arbitrary state. In order to enable generalization over nearby states and avoid the need to explore every existing state in the state space, a linear function approximation scheme with a nonlinear feature projection using tile coding is employed.

$$V_\pi(s) \approx \hat{V}_\pi(s; \mathbf{w}) = \psi(s)^T \mathbf{w}$$

Where $\psi(s)$ denotes the projection of the tile coding and $\mathbf{w}$ denotes the function weights of the linear approximator. Optimizing the function weights can be realized using gradient descent on the squared error $\mathcal{L}_{SE}$ between the actual state-value and the approximation.

$$\mathcal{L}_{SE}(s, \mathbf{w}) = \left(V_\pi(s) - \hat{V}_\pi(s; \mathbf{w})\right)^2$$
$$\nabla_{\mathbf{w}} \mathcal{L}_{SE}(s, \mathbf{w}) \propto (V_\pi(s) - \hat{V}_\pi(s; \mathbf{w})) \nabla_{\mathbf{w}} \hat{V}_\pi(s; \mathbf{w})$$

Since the true state-value $V_\pi(s)$ is unknown, it is replaced using bootstrapping, as typical for the TD learning procedure. A complete optimization step with learning rate η_w uses the TD error δ as follows.

$$\delta = r_t + \gamma \hat{V}_\pi(s_{t+1}; \mathbf{w}_t) - \hat{V}_\pi(s_t; \mathbf{w}_t)$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta_w \delta \nabla_{\mathbf{w}} \hat{V}_\pi(s_t; \mathbf{w}_t) = \mathbf{w}_t + \eta_w \delta \psi(s_t)$$

This method learns approximating the long term discounted reward of a given state, when following a certain policy π, but it does not provide inference about taking a certain action. Therefore, a parameterized stochastic policy is defined, that is then successively trained using the learned value estimate. The policy π_θ, parameterized by θ, is defined as a normalized exponential function (softmax) over action preferences $H(s, a; \theta)$. The action preference is estimated using a linear function approximation and employing a feature mapping in form of tile coding.

$$\pi_\theta(a|s) = \frac{e^{H(s,a;\theta)}}{\sum_{b \in \mathcal{A}} e^{H(s,a;\theta)}}$$

$$\text{with} \quad H(s, a; \theta) = \phi(s, a)^T \theta$$

$\phi(s, a)$ signifies the feature encoding for a state-action pair in form of tile coding and θ represents the function weights that need to be optimized. For policy optimization, the derivative of the log-probabilities given by the policy is needed.

$$\nabla_\theta \log \pi_\theta(a|s) = \phi(s,a) - \sum_{b \in \mathcal{A}} \pi_\theta(b|s)\, \phi(s,b)$$

The policy gradient is used for the optimization of the parameters θ with learning rate η_θ.

$$\theta_{t+1} = \theta_t + \eta_\theta A_\pi(s,a) \nabla_\theta \log \pi_\theta(a|s)$$

The advantage $A_\pi(s,a)$ is estimated by the state-value TD-error as previously described. Alternatively to using only the policy gradient of the current step, eligibility traces accumulate the gradients over consecutive learning steps, improving learning stability and sensitivity for delayed reward.

With the two given components, an actor-critic learning algorithm can be implemented. Since a good state-value estimate is needed to provide meaningful guidance for the actor, usually the critic learns at a higher rate than the actor, which will be achieved by tuning the learning rates, such that $\eta_w > \eta_\theta$. An outline for the resulting learning procedure is given in Alg. 1.

Alg. 1: Actor-Critic algorithm for radio parameter optimization

$\eta_w, \eta_\theta \in (0,1)$ ▷ Initialize learning rates
$\gamma \in (0,1), \lambda \in (0,\gamma)$ ▷ Initialize discount factors
$\mathbf{w}, \theta = \mathbf{0}$ ▷ Initialize function parameters
$D = \mathbf{0}$ ▷ Initialize eligibility traces
$s_1 \in \mathcal{S}$ ▷ Initialize starting state
for $t = 1, \dots, T$ **do**
  $a_t \sim \pi_\theta(a|s_t)$ ▷ Sample action from Policy
  $s_{t+1} \sim P(s|s_t, a_t)$ ▷ Observe successor state
  $r_t = R(s_t, s_{t+1}, a_t)$ ▷ Receive reward
  $\delta = r_t + \gamma \hat{V}_\pi(s_{t+1}; \mathbf{w}) - \hat{V}_\pi(s_t; \mathbf{w})$ ▷ Compute TD-error
  $\mathbf{w} \leftarrow \mathbf{w} + \eta_w \delta \psi(s)$ ▷ Optimization state-value
  $D \leftarrow \lambda D + \nabla_\theta \log \pi_\theta(a_t|s_t)$ ▷ Update eligibility traces
  $\theta \leftarrow \theta + \eta_\theta \delta D$ ▷ Optimization policy
end

Implementation and Preliminaries

The method described has been developed to enable learning on low-power end-devices with constrained computing capabilities, such as sensor nodes. In order to evaluate the algorithms, they were implemented on real hardware and experimentally tested under real world conditions. The hardware platform chosen for experimentation is based on the *STM32WL5x* system-on-chip (SoC), that embeds a low-power Cortex-M4 microcontroller and a sub-GHz radio based on *Semtech's* SX126x.

During the experiments two radio modules are used, one takes the role of an end-node that sends messages to the second module acting as the receiving node or base station. Unlike dedicated gateway radio chips, the used wireless module cannot listen on different channels simultaneously. Thus, the radio configuration for an upcoming message transmission is communicated with the previous transmission. Feedback for a transmission is sent over the same channel as the message. If a transmission fails, the last successfully used channel is used. In case this fails due to dynamic influences, a backup channel that is defined as highest spreading factor (SF12) with highest transmission power (14 dBm) is used.

The computation of the reward for the RL algorithm is based on the required energy for a transmission. This is calculated using a simple energy model consisting of a product of the real transmission power P_{TX} and the time on air T_{OA} of the transmitter. The real transmission power was measured with a digital multi meter and stored on the microcontroller as a lookup table for every configurable transmission power. The time on air can be computed using a formula based on the used radio configuration [24].

The function approximation scheme to estimate the state-value $\hat{V}(s;\mathbf{w})$ and the action preference $H(s,a;\theta)$ both use tile coding to encode the state features. The number of tilings, their offset and size of partitions need to be defined. These hyperparameters set the trade-off between generalization over similar states and accuracy of feature representation. The used values are given in tab. 1.

Tab. 1: Parameters for encoding of state variables using tile coding with 5 tilings

State Variable	# Tiles	Offset per Tile
SF	3	1
TP	4	2
RSSI	4	5
SNR	4	4

In case of the RSSI and SNR value, one tile is reserved for the event of a failed transmission. If this happens, the offset is ignored, thus, the same tile is activated in all tilings. In case of the action preference $H(s,a;\theta)$ the state-action pair has to be encoded. Since there are $|\mathcal{A}| = 5$ discrete actions, each action uses a separate state encoding.

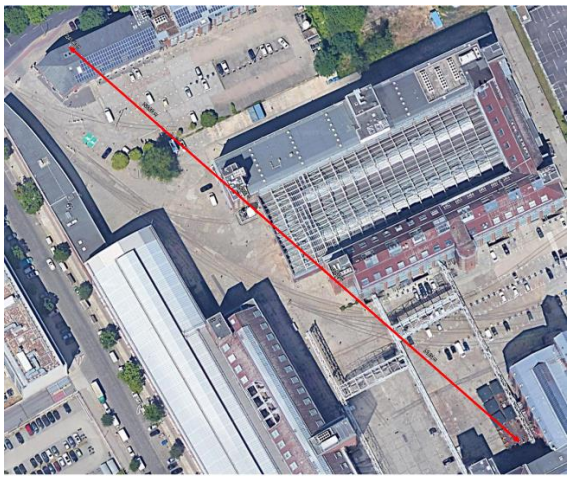

Fig. 1. Satellite view of the urban setting for communication. Distance between transmitter and receiver is 241 m. (©2023 CNES/Airbus, GeoBasis-DE/BKG,GeoContent,Maxar Technologies, Karten-daten ©2023)

The goal of the algorithm is to identify near optimal parameter selection to save energy while ensuring communication reliability. To reflect the importance of the long term energy savings the discount factor is set to $\gamma = 0.99$, this mean the agent tries to maximize its long term future reward. The discount factor for eligibility traces is set to $\lambda = 0.9\,\gamma$. The learning rates are set to $\eta_w = 0.6$ and $\eta_\theta = 0.4$ for the critic and policy respectively.

To accurately assess the energy demand, it is important to account for both the transmission energy E_{TX} and the energy overhead E_{Opt} caused by additional computations on the end node. The time for sampling actions based on the policy function is insignificant, optimizing the value weights takes 4 ms per step and updating the policy parameters takes 21 ms, this results in a total time for optimization of 25 ms per step. At 48 MHz clock speed and supply voltage of 3.3 V the controller consumes 3.5 mA.

Experimental Comparison with ADR in LoRaWAN

The ADR scheme is the standard procedure for data rate and parameter optimization in *LoRaWAN*. There exists no specification how ADR has to be implemented but *Semtech* gives a recommendation for a baseline algorithm [25] that is used during this experiment. Although, *LoRaWAN* It will be compared experimentally to the developed method described in this work, mainly in terms of long term energy demand. The setting of the experiment is in an urban environment, an overview is shown in Fig. 1. The distance between the receiver and transmitter are 241 m. The walls of the building, cars and other objects in between the two radio transceivers cause some varying channel degradation.

In the experimental setting, the transmitter sends 5,000 packets of 20 bytes each using the radio parameters proposed by the optimization algorithm. The parameters are recorded and evaluated after the experiments. Fig. 2 shows the accumulated energy consumption after each transmission for the experiment. It includes the energy for transmission itself and the consumption induced by the processing overhead. Five runs of optimization were analyzed, the plot shows the range between the least and most energy efficient run, as well as the mean over the runs. It can be seen that the ADR mechanism in *LoRaWAN* quickly determines a radio configuration after a short probing phase and keeps it throughout the whole duration, which results in a linear energy consumption over time. In different runs, ADR used slightly different data rates, which results in a difference in energy consumption over time. In case of the implemented actor-critic RL algorithm, the transmitter behaves randomly and seems less goal oriented in the beginning. It needs many samples (transmissions) until a stable configuration for the transmitter is reached and kept. However, due to the long lifetime of wireless sensor systems, even small energy savings scale over time and may accumulate to long term savings.

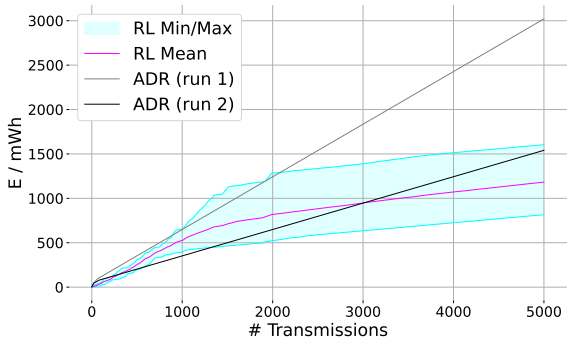

Fig. 2. Cumulative energy consumption of the two methods for comparison in urban setting. ADR converges to a reliable setting quickly but the utilized configuration differs in the two runs. The RL based method need more trial and error but eventually converge to a more efficient configuration.

Comparing the energy consumption with the more efficient ADR run, the RL based approach breaks even at around 3000 message transmissions on average. If an application with one transmission every 15 minutes is assumed, this would mean the energy demand breaks even at around 31 days, which is usually only a fraction of the desired lifetime of a wireless sensor.

Adaption to a Dynamic Wireless Channel

ADR in *LoRaWAN* is not supposed to be used in dynamic channels, e.g. when the transmitting unit is changing its position over time. The RL algorithm supports decision-making by locally incorporating contextual information to determine the appropriate transmitter configuration for the next operation. This means, channel degradation or improvement can be monitored in real-time and addressed by adjusting the radio configuration adaptively. This creates opportunities for mobile sensor nodes and enables the exploration of new network topologies. The end nodes are not dependent on the network server for parameter provisioning, thus, improving their capability of communicating effectively with each other, which is essential in multi-hop or meshed networks.

To demonstrate this behavior the node is set to two different locations where it learns to adapt to the respective condition, the first channel is set to be worse than the second (greater distance to the receiver). The agent is left to converge on the first position for 2000 steps, then, the position is changed to the second location. It can be observed, that the process of finding efficient parameters is quicker than in the first case, showcasing some degree of generalization. After 2000 initial transmissions on each channel, the position is switched back and forth two more times, with 1000 transmissions on each channel. The time for adaption to the new channel is accelerated each time, the agent observes a changed channel condition. This observation can be depicted by comparing the average energy consumption during the process, which is shown in fig. 3. Note, that the energy consumption in the second channel is so much smaller, that it appears to be zero in parts of the plot.

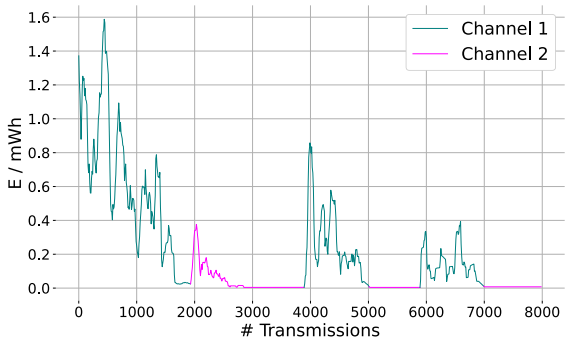

Fig. 3. Energy consumption of the RL method in an alternating channel. Process of finding efficient parameters for the changed channel is faster in every iteration. Energy consumption averaged over 10 steps for smoother visualization.

Conclusion

This work presents an approach to enable low-power sensor nodes to utilize algorithms from the field of reinforcement learning. This opens opportunities across a wide range of application domains, exemplified here by applying it to radio parameter optimization. Other conceivable use-cases include tasks such as network routing, or timing and balancing of data acquisition, processing, and communication. Moreover, the algorithm's versatility allows for adaptable optimization across diverse objectives, such as maximizing data throughput, enhancing energy efficiency, and ensuring robustness against failures. Decentralized organization in general and adaptive mechanisms such as self-healing and self-configuring capabilities are desired properties of wireless sensor systems in the future. These needs highlight the appeal of intelligent behavior of sensor nodes in the context of wireless sensor networks, which marks a paradigm shift as learning algorithms are commonly associated with substantial computational power typically found in cloud servers.

The implementation of RL algorithms has been described with a special focus on resource constrained hardware as it is common for low-power sensor devices. The outlined algorithm was customized with the goal of developing a decentralized and adaptive learning algorithm that aims at optimizing energy demand of WSNs.

The developed method was implemented and its performance was experimentally evaluated in comparison to a state-of-the-art concept for parameter optimization in the networking standard *LoRaWAN*. Despite the approach of decision making on the device, the presented algorithms showed to be able to compete in a realistic experimental setting. The ability of saving a significant amount of energy long-term helps maximizing the lifespan of sensor nodes in a network. However, ADR doesn't need the amount of downlink communication in direction of the end-node, which serves its purpose in large star networks where a single gateway services thousands of nodes. Locally optimizing radio configurations in real-time opens up opportunities for exploring different network topologies for *LoRa* networks and may help improving overall network efficiency. The method showed promising results in the experimental setup. Nevertheless, it will be a part of future research to further evaluate its performance based on diverse and long-term deployments.

References

[1] A. Jay, Number of Internet of Things (IoT) Connected Devices Worldwide 2022/2023: Break-downs, Growth & Predictions, *FinancesOnline*, (Access: 20.05.2024) https://financesonline.com/number-of-internet-of-things-connected-devices

[2] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A Survey on Application of Machine Learning for Internet of Things*, International Journal of Machine Learning and Cybernetics 9* (2018); doi:10.1007/s13042-018-0834-5

[3] X. Wang, H. Chen, S. Li, A Reinforcement Learn-ing-Based Sleep Scheduling Algorithm for Compres-sive Data Gathering in Wireless Sensor Networks, *EURASIP Journal on Wireless Communications and Networking* (2023); doi:10.1186/s13638-023-02237-4

[4] H. Zhang, Y. Song, M. Yang, Q. Jia, Modeling and Optimization of LoRa Networks under Multiple Constraints, *Sensors* 23 (18) (2023); doi: 10.3390/s23187783

[5] A. Farhad, J.-Y. Pyun, LoRaWAN Meets ML: A Survey on Enhancing Performance with Machine Learning, *Sensors* 23 (15) (2023); doi: 10.3390/s23156851

[6] M. González-Palacio, D. Tobón-Vallejo, L. M. Sepúlveda-Cano, S. Rúa, Machine-Learning-Based Combined Path Loss and Shadowing Model in LoRaWAN for Energy Efficiency Enhancement, *IEEE Internet of Things Journal* 10 (12) (2023); doi: 10.1109/JIOT.2023.3239827

[7] F. Azizi, B. Tyemuri, R. Aslani, M. Rasti, J. Tolvaneny, P. H. J. Nardelli, MIX-MAB: Reinforcement Learning-based Resource Allocation Algorithm for LoRaWAN, *2022 IEEE 95th Vehicular Technology Conference* (2022); doi: 10.1109/VTC2022-Spring54318.2022.9860807

[8] Sensors (Open Access journal): Intelligent Sensors (Zugriff: 15.05.2024) https://www.mdpi.com/journal/sensors/sections/Intelligent_Sensors

[9] F. Fraternali, Towards Large-Scale Autonomous Wireless Sensor Networks, *arxiv:1906.12001* (2019)

[10] C. Yan, Q. Ji-hong, Application Analysis of Complex Adaptive Systems for WSN, *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)* 7, S. 328-331 (2010); doi: 10.1109/ICCASM.2010.5620113

[11] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing, *Proceedings of the IEEE 2019*, 107 (8), 1738–1762 (2019); doi: 10.1109/JPROC.2019.2918951

[12] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, *The MIT Press, Cambridge, Massachusetts*, second edition (2018)

[13] S. P. Singh, T. Jaakkola, M. I. Jordan, Learning Without State-Estimation in Partially Observable Markovian Decision Processes, *Machine Learning Proceedings 1994*, 284–292 (1994); doi: 10.1016/B978-1-55860-335-6.50042-8

[14] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning*, 8(3-4), 229–256 (1992); doi: 10.1023/A:1022672621406

[15] V. R. Konda, J. N. Tsitsiklis, OnActor-Critic Algorithms, *SIAM Journal on Control and Optimization*, 42(4),1143–1166 (2003); doi: 10.1137/S0363012901385691

[16] S. P. Singh, R. S. Sutton, Reinforcement Learning with Replacing Eligibility Traces, *Machine Learning*, 22(1/2/3), 123–158 (1996); doi: 10.1007/BF00114726

[17] M. G. Lagoudakis, Value Function Approximation, *Encyclopedia of Machine Learning and Data Mining*, 1311–1323, Springer US, Boston (2017); doi: 10.1007/978-1-4899-7687-1_876

[18] A. A. Sherstov, P. Stone, Function Approximation via Tile Coding: Automating Parameter Choice, *Abstraction, Reformulation and Approximation*, volume 3607 of Lecture Notes in Computer Science, 194–205,Springer, Berlin, Heidelberg (2005); doi: 10.1007/11527862_14

[19] A. Bensky, Communication Protocols and Modulation, *Short-range Wireless Communication*; 85–127, Elsevier (2019); doi: 10.1016/B978-0-12-815405-2.00004-X

[20] Semtech Corporation: AN1200.22 LoRa Modulation Basics. *Semtech Corporation* (2015)

[21] M. Bor, U. Roedig, LoRa Transmission Parameter Selection, *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS),* 27–34, Ottawa, ON (2017); doi: 10.1109/DCOSS.2017.10

[22] Semtech Corporation: Understanding the LoRa Adaptive Data Rate, *Semtech Corporation* (2019)

[23] R. Serati, B. Teymuri, N. A. Anagnostopoulos, M. Rasti, ADR-Lite: A Low-Complexity Adaptive Data Rate Scheme for the LoRa Network, *arXiv: 2210.14583* (2022); doi: 10.48550/arXiv.2210.14583

[24] Semtech Corporation: AN1200.13 SX1272/3/6/7/8: LoRa Modem Designer's Guide, *Semtech Corporation* (2013)

[25] Semtech Corporation: LoRaWAN – simple rate adaptation recommended algorithm, *Semtech Corporation* (2016)