Energy Aware Deep Reinforcement Learning Scheduling for Sensors Correlated in Time and Space

Jernej Hribar, Andrei Marinescu, Alessandro Chiumento, and Luiz A. DaSilva

***Abstract*—Millions of battery-powered sensors deployed for monitoring purposes in a multitude of scenarios, e.g., agriculture, smart cities, industry, etc., require energy-efficient solutions to prolong their lifetime. When these sensors observe a phenomenon distributed in space and evolving in time, it is expected that collected observations will be correlated in time and space. This paper proposes a Deep Reinforcement Learning (DRL) based scheduling mechanism capable of taking advantage of correlated information. The designed solution employs Deep Deterministic Policy Gradient (DDPG) algorithm. The proposed mechanism can determine the frequency with which sensors should transmit their updates, to ensure accurate collection of observations, while simultaneously considering the energy available. The solution is evaluated with multiple datasets containing environmental observations obtained in multiple real deployments. The real observations are leveraged to model the environment with which the mechanism interacts as realistically as possible. The proposed solution can significantly extend the sensors' lifetime and is compared to an idealized, all-knowing scheduler to demonstrate that its performance is near-optimal. Additionally, the results highlight the unique feature of proposed design, energy-awareness, by displaying the impact of sensors' energy levels on the frequency of updates.**

***Index Terms*—Deep Reinforcement Learning, Reinforcement Learning, Low-Power Sensors, Internet of Things**

I. Introduction

Millions of low-power devices are being deployed to provide services in smart cities [1], Industry 4.0 [2], smart agriculture [3], and other Internet of Things (IoT) applications. Many of these devices are low-cost sensors powered by non-rechargeable batteries. Their role is to provide sensed information to services, which use this information to make decisions. For example, in smart agriculture, a service controlling an irrigation system requires information from various sensors to decide which fields to water. The main challenge is to provide accurate and up-to-date information to services while keeping the deployments of battery-powered devices functional for as long as possible.

J. Hribar is with CONNECT, Trinity College Dublin, Ireland. Email: jhribar@tcd.ie.

A. Marinescu is with Eaton's Center for Intelligent Power, Dublin, Ireland. Email: AndreiMarinescu@eaton.com.

A. Chiumento is with the University of Twente, The Netherlands, Email: a.chiumento@utwente.nl.

and L. A. DaSilva is with the Commonwealth Cyber Initiative, Virginia Tech, USA. Email: ldasilva@vt.edu.

This work was funded in part by the European Regional Development Fund through the SFI Research Centres Programme under Grant No. 13/RC/2077 SFI CONNECT and by the SFI-NSFC Partnership Programme Grant Number 17/NSFC/5224. The corresponding author is Jernej Hribar.

In a system of multiple sensing devices observing the same physical phenomenon, it is expected that the information collected will be correlated in time and space. By relying on this correlation, we have shown that it is possible to increase the time between consecutive updates by each sensor, thereby increasing its lifetime, without compromising the accuracy of the information provided to the IoT service [4]. In the absence of up-to-date information from one sensor, the system can rely on more recent information obtained from a correlated sensor. In this paper, we propose a Deep Reinforcement Learning (DRL)-based scheduling mechanism capable of determining how frequently each low-power sensor should transmit its observations so as to furnish the service with accurate information while maximising the lifetime of the network.

We consider an IoT system where low-power sensors transmit periodic updates to a gateway. The gateway is able to schedule when the next update by each sensor should occur. The gateway relies on a data-driven approach to make this determination, by considering the energy available to each low-power sensor and the need for fresh updates, according to concepts related to the Age of Information (AoI) [5]. We design a DRL solution capable of learning from past experiences.

Multiple, often non-trivially connected, factors impact the decision of when a particular low-power sensor should transmit new information. These factors can be external to the sensor, such as whether a nearby sensor has recently transmitted updated information, changes in the observed physical phenomenon, etc., or internal to the sensor, e.g., the remaining energy, transmission power, or location. The use of DRL enables us to design a scheduling mechanism capable of determining when sensors should transmit updated sensed information to a gateway, by weighing all relevant factors to make an efficient decision.

In this paper, we use DRL to conserve battery-powered sensors' energy by leveraging the correlation exhibited in the information they collect. In particular, we make the following contributions:

- The main contribution of this paper is the design of a DRL energy-aware scheduler that is capable of determining when an IoT sensor should transmit its next observation. We make use of Deep Deterministic Policy Gradient (DDPG), a DRL algorithm, to arrive at an efficient transmission schedule for sensors based on their available energy, the freshness of the information collected, and the expected lifetime of all other sensors

in the network, without compromising the accuracy of the information delivered to the application.
- A unique feature of our solution is energy balancing. Our mechanism is capable of determining to what extent the energy available to one sensor can be used to prolong the lifetime of others. We benchmark our solution by comparing it to an ideal scheduler that acts as an oracle and is assumed to know the ground truth about the phenomenon being observed.
- To validate our solution, we use more than five different datasets to demonstrate its near-optimal performance in a variety of scenarios. Note that we leverage real data to model the environment in which our scheduler operates as realistically as possible.

After briefly reviewing the relevant literature in Section II, in Section III we describe how a system of sensors collecting correlated information can estimate the accuracy of their observations. We describe the decision-making problem that our proposed scheduling mechanism is capable of assisting with in Section IV. We present how we incorporated DDPG into our proposed mechanism (Section V A), describe the system dynamics using states, actions, and rewards from a Reinforcement Learning (RL) perspective (Section V B), and provide implementation details (Section V C). We utilize data obtained from real deployments to show that the learned behaviour significantly prolongs the sensors' lifetime and achieves near-optimal performance (Section VI A). Additionally, we demonstrate the scheduling mechanism's energy awareness when deciding on the sensors' transmission times (Section VI B). Finally, we summarise our main findings and discuss our future work in Section VII.

II. Related Work

Our work leverages the recently proposed AoI metric [6], which quantifies the freshness of information. The AoI metric measures the time since a status update generated by a source, e.g., a sensor, was last received. Each status update contains the latest information collected by the source and a timestamp. The more recently the status update was generated, the more relevant it is to the decision process. Finding the optimal update rates with which sources should send information is non-trivial [7], and considering correlation between status updates adds to the complexity of the problem, as an update from one source lowers the requirement for fresh information on all other correlated sources. We were the first to propose taking advantage of fresher information from correlated sources to conserve the energy of battery-powered sources by prolonging the times between sources' consecutive updates [4]. Subsequently, the authors in [8] and [9] analyzed from different perspectives a system with correlated sources. In [8], the authors consider a system in which multiple sources may observe the same information captured by sensors, i.e., multiple sources can obtain the same status update. In [9], the authors considered a one-dimensional static random field from which multiple sources transmitted observations. The works in [4], [8], [9] considered only the impact of correlation on the timeliness of information to establish the desired frequency of updates. This paper presents a solution that considers both the timeliness of the information and the energy available to the sources to arrive at a scheduling of information updates that prolongs the lifetime of battery-powered sensors.

Our work is also related to, but differs in crucial aspects from, the various approaches proposed for energy efficiency in the context of wireless sensor networks (WSNs) [10]–[14]. Most proposed works for WSNs rely on detection or reconstruction of the observed phenomena, through data prediction or model-based active sampling methods, to improve the low-power sensors' energy efficiency. In contrast, we focus on the timeliness of updates, and in particular the AoI, and then employ RL to determine how to utilize correlated measurements to reduce the rate at which sources transmit their updates.

RL has been applied to other energy-aware networking solutions [15]–[24]. For example, in [16] the authors use Q-learning to enhance the spectrum utilization of industrial IoT devices. They demonstrate that devices are capable of learning a channel selection policy to avoid collisions, thus reducing retransmissions. The authors in [22] propose a channel-state aware scheduling mechanism using a Deep Q-Network (DQN) that enables a device to learn when to transmit and which channel to use for transmission. The authors in [17] investigate the use of a semi-supervised DQN to improve indoor users' location estimation by leveraging information on Bluetooth signal strength. The work in [19] relies on a no regret RL approach, while the authors in [20] apply an actor-critic algorithm to analyse how an energy-harvesting device will collect energy and schedule its transmissions accordingly. In both cases, the objective is more effective power management, using RL to prevent power outages, i.e., to avoid the situation where an energy-harvesting device completely depletes its energy. The power control in an energy harvesting system is also investigated in [24]. The authors, using a multi-agent approach (with a DQN algorithm), designed a distributed power control mechanism that maximises devices' throughput. The authors in [23] use a DQN algorithm to solve a task offloading problem for vehicles. Their solution is capable of saving energy by selecting more efficiently where in the edge a task from a vehicle should be processed. In [21], the authors apply a deep belief neural network to design a data recovery mechanism for sensors that collect spatio-temporally correlated information. Their mechanism is capable of determining which observations from other sensors could be used to replace missing or corrupted observations.

In [25], we first introduced a variant of the proposed scheduling mechanism. In this paper, we extend our work by designing an improved DRL-based mechanism (based on a DDPG algorithm, while [25] relied on a DQN algorithm) and providing a comprehensive evaluation of the effectiveness of our solution. We compare our solution to one conventional way of setting the sensors' update intervals, and to the optimal scheduler we designed specifically for evaluation purposes. Furthermore, we test our mechanism over three additional real-world datasets. Another extension of our work is in providing new results that demonstrate the energy balancing aspect of our solution.

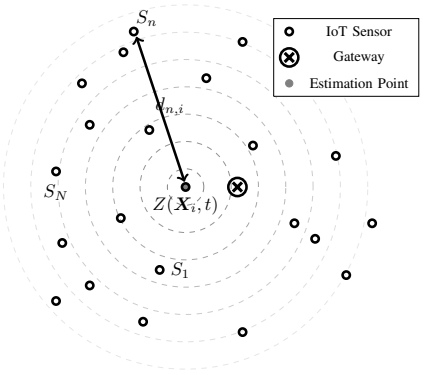

Fig. 1: A system of N randomly distributed sensor nodes, whose observations are used to estimate the value of observed physical phenomenon, $Z(\boldsymbol{X},t)$, at location $\boldsymbol{X}_i$ at time t.

III. Quantifying the accuracy of observations

We consider a sensor network with N geographically distributed sensors transmitting observations to a gateway for collection. The main purpose of these sensors, denoted as $\{S_1,\ldots,S_N\}$, is to observe a physical phenomenon $Z(\boldsymbol{X},t)$ distributed in space $\boldsymbol{X}$ and evolving in time t. In our work, we perform the physical modeling of the observed phenomenon using observations obtained in a real IoT deployment [26], [27]. Sensors are deployed at positions $\boldsymbol{X}_n$ and transmit periodic observations with an update interval $T_n, n=1,\ldots,N$. In our system, we assume that the latest received observation from a sensor replaces the previously received information as, according to the AoI paradigm, the freshest information is the most relevant in the decision making process [6]. The AoI metric measures the time elapsed since the sink, i.e., a gateway, received a new observation from the sensor. We denote the AoI with $\Delta_n(t)$. Whenever the system receives an observation from location $\boldsymbol{X}_n$ at time t_n, the system will anticipate the arrival of the next observation from location $\boldsymbol{X}_n$ at time instance $t=t_n+T_n$. Additionally, the value of AoI will drop to zero. Meaning, that the value of AoI is limited to an interval between 0 and T_n. We write the collected observations into a vector $\mathbf{Y}=[y_1,\ldots,y_N]^T$ with $y_n=Z(\boldsymbol{X}_n,t_n)$ where t_n is the latest time at which sensor n has reported an observation.

The system can estimate the value of the observed physical phenomenon at the desired location $\boldsymbol{X}_i$ at any time instant t using the collected information, as presented in Fig 1. We denote the Euclidean distance between sensor S_n and the location of interest $\boldsymbol{X}_i$ as $d_{n,i}$. With $\Delta_{n,i}(t)$ we denote the time elapsed since the system received the latest observation from sensor S_n, i.e., the AoI, $\Delta_n(t)\coloneqq t-t_n$. To estimate the observed process we apply a Linear Minimum Mean Square Error (LMMSE) estimator which is commonly used for such problems as demonstrated in [28]. One of the biggest advantage of using LMMSE is that it provides a mathematically tractable solution and to obtain it, we only require the expected values, variances, and the covariance. In our system, all three are available. Therefore, we can approximate the value of the observed physical phenomenon at position $\boldsymbol{X}_i$ at time instant t, as:

$$\hat{y}_i(t)=\sum_{n=1}^{N} w_{n,i}(t)y_n, \tag{1}$$

where $w_n(t), n=0,\ldots,N$ are LMMSE estimator weights.

Following the analysis in [29], we obtain the LMMSE estimator weight vector $\boldsymbol{W}_i(t)=[w_{0,i}(t),\ldots,w_{N,i}(t)]^T$ as follows:

$$\boldsymbol{W}_i(t)=\Big(\boldsymbol{C_{YY}}(t)\Big)^{-1}\boldsymbol{c}_{i,\boldsymbol{Y}Z}(t). \tag{2}$$

The matrices $\boldsymbol{C_{YY}}(t),\boldsymbol{c}_{i,\boldsymbol{Y}Z}(t)$ are covariance matrices, required to determine $\boldsymbol{W}_i(t)$:

$$\boldsymbol{C_{YY}}(t)=\begin{bmatrix}\mathrm{C}_{1,1}(t)\ldots\mathrm{C}_{1,N}(t)\\ \vdots\qquad\quad\vdots\\ \mathrm{C}_{N,1}(t)\ldots\mathrm{C}_{N,N}(t)\end{bmatrix};\boldsymbol{c}_{i,\boldsymbol{Y}Z}(t)=\begin{bmatrix}\mathrm{C}_{1,i}(t)\\ \vdots\\ \mathrm{C}_{N,i}(t)\end{bmatrix}; \tag{3}$$

in which $\mathrm{C}_{j,k}(t); j,k=1,\ldots,N$, is the covariance of observations y_j and y_k, and $\mathrm{C}_{j,i}(t)$ is the covariance of y_j and the observed process Z at the desired location of the estimation. To obtain the required matrices we can rely on a covariance model and utilize past observations to determine its values. We adopt a separable covariance model defined in [30]. With it we model how observations collected at different instants in time and different locations relate to each other. We express the covariance between two observations or one observation and the estimation point, with AoI difference $\Delta_{j,k}(t)$ and distance $d_{j,k}$ apart as:

$$\mathrm{C}_{j,k}(d_{j,k},t|\theta_1(t),\theta_2(t))=exp(-\theta_2(t)d_{j,k}-\theta_1(t)\Delta_{j,k}(t)). \tag{4}$$

Note that $\theta_1(t)$ and $\theta_2(t)$ are scaling parameters of time and space, respectively. With $d_{j,k}$ we denote the Euclidean distance between sensor S_j and the location at which the system estimates the value of the observed physical phenomenon, or between sensor S_j and sensor S_k. Both scaling parameters change over time and are extracted from the obtained observations. In our work, we follow a scaling extraction method with Pearson's correlation coefficient formula for samples, as described in [31].

The selected covariance function provides a good fit to model spatial and temporal variation for many physical phenomena. For example, in [30], the authors showed that such a covariance model could be applied to wind-speed data. We demonstrated in [4] that the selected model is applicable to the temperature and humidity sensor data used in the evaluation section. Additionally, such spatio-temporal correlation can be observed in many IoT sensor deployments: examples include IoT systems in a smart city measuring air pollution, precipitation, or noise [1], and smart farm applications in which an IoT system monitors soil parameters [3].

Every time the system employs Eq. (1) to estimate the value of the observed physical phenomenon it makes an error. By using matrices $\boldsymbol{C_{YY}}(t)$ and $\boldsymbol{c}_{\boldsymbol{Y}Z}(t)$ it is possible to determine the Mean Square Error (MSE) in the estimation as:

$$\varepsilon_i(\boldsymbol{X}_i, t|\theta_1(t), \theta_2(t)) = \sigma_Z^2 - \boldsymbol{c}_{\boldsymbol{ZY}}(t)\boldsymbol{W}_i(t), \quad (5)$$

where $\boldsymbol{c}_{\boldsymbol{ZY}}$ is the transpose of $\boldsymbol{c}_{\boldsymbol{YZ}}$ defined above, and σ_Z^2 represents the variance of the observed phenomenon. The estimation error provides a measure with which the gateway can quantify the quality of the information currently provided by the sensing process: the lower the value of the estimation error, the more accurate the estimated values and the lower the need for an additional update. Hence, by measuring the average estimation error between two consecutive updates the gateway can assess how accurate and up-to-date the observations collected by the system are. In our work, we control the accuracy of our sensing process by setting as a constraint the maximum MSE of the estimator, ε^*. In short, the purpose of our proposed updating mechanism is to set sensors' update intervals in such a way that the average estimation error will not exceed the set target. In the next section, we describe the optimisation problem that the gateway must solve: maximising the network lifetime, constrained by the target accuracy in the measurements.

IV. Problem Formulation

The scenario of interest to our work is the use of inexpensive battery-powered sensors, transmitting observations to a gateway for collection. The gateway aims to schedule the transmission of observations in such a way that the accuracy of the information collected will satisfy the service requirements, i.e., average estimation error below ε^*, while, simultaneously, trying to prolong the lifetime of the deployed sensor network. Whenever a sensor transmits an observation, the updating mechanism residing in the gateway decides on the low-power sensors' next update time by evaluating the accuracy of collected observations and the sensors' available energy. Each sensor's lifetime depends on the frequency of transmitted observations, i.e., the time between two consecutive updates, and on the continuous power consumption, which is independent of transmissions.

In this work, we assume that a non-rechargeable primary battery powers the low-power sensors. In such a case, the energy consumption depends on how often a sensor transmits an observation and the energy it needs to function regardless of the mode of operation. Therefore, we can model a sensor's lifetime $\mathcal{L}_n(T_n)$ as in [32]:

$$\mathcal{L}_n(T_n) = \frac{E_0}{P_c + \frac{E_{tr}}{T_n}}, \quad (6)$$

where E_0 represents the sensors' starting energy and P_c is the continuous power consumption, and E_{tr} represents the energy required to acquire and transmit the observation. The continuous power consumption is the power that the sensor always consumes, for example, leakage current, and depends solely on the sensor hardware components. For low-power IoT sensors, P_c is in the range of a few μW or less. The energy required to transmit the observation, i.e., E_{tr}, depends on many factors such as the size of the transmitted packet, the energy required to take the measurement, and channel conditions.

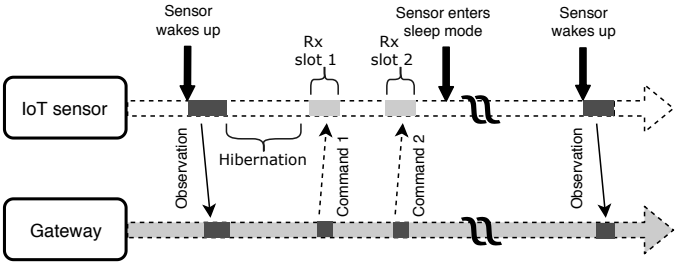

Fig. 2: Message sequence of a low-power sensor using LoRaWAN.

Energy is not the only factor the updating mechanism has to take into account. As described in [33], low-power sensors are also constrained in terms of available computing power, memory, communication capabilities, etc. Limited processing power and memory prevent the use of a sophisticated algorithm on the sensor itself. Therefore, computationally demanding tasks when making a decision should be carried out at the gateway. Additionally, these sensors rely on low data rate transmission, meaning that communication messages between sensors and gateway should be kept to a minimum. Furthermore, to extend their lifetime, low-power sensors rely on the use of sleep mode. When a sensor is in sleep mode, the rest of the network cannot communicate with it. Consequently, the gateway has to inform each sensor, while the sensor is still in active mode, when it should wake up again and transmit the next observation. Sleep mode is supported by most Low-Power Wide-Area Network (LPWAN) standards, such as SigFox, Weightless, Long Range Wide Area Network (LoRaWAN)[1], and Narrowband IoT (NB-IoT) [34]. The low-power sensor is usually in active mode only after it has transmitted. For example, a sensor using a LoRaWAN class A radio will listen for two short time-windows after it has transmitted, as illustrated in the LoRaWAN message sequence in Fig. 2 [35], meaning that the updating mechanism only has a short time-window to provide a response.

The gateway's goal is to prolong the network lifetime. We define the network lifetime as the lifetime of the sensor with the shortest lifespan in the deployment. In other words, the network lifetime expires the moment one sensor depletes all of its energy. To that end, the gateway should aim to minimise transmissions by all sensors, i.e., increase T_n, and, when updates are required, favour sensors with higher remaining energy, all while keeping the average estimation error of the observed physical phenomenon at every location of interest below the pre-specified value, i.e., ε^*. In a real deployment, services dictate which locations are of interest for the system. In this paper, we consider every sensor location, i.e., $\boldsymbol{X}_n$, to be a location of interest, meaning that system has to make accurate estimations at the location of every sensor while keeping sensors' power consumption to a minimum. We summarise the decision-making process in Fig. 3. The gateway decides on each sensor's next update time by evaluating the accuracy of the collected observation and the sensors' available

[1]For more information, the reader may visit http://www.sigfox.com; http://weightless.org; http://www.lora-aliance.org, respectively.

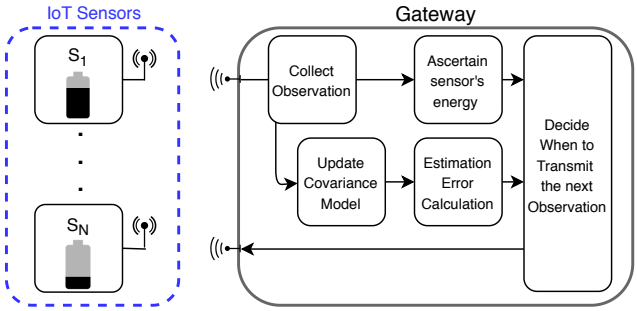

Fig. 3: High level overview of the decision making process in the gateway.

energy, which it can determine from the sensor's reported power supply measurement. The gateway can then decide when the sensor should transmit its next observation.

Transmission from a sensor $\hat{n}$ starts the decision-making process at time $t_{\hat{n}}$: due to the energy-constrained nature of the devices, the gateway has a limited time to determine the sensor's next update time, i.e., determine for sensor $\hat{n}$ a new sensor update interval $T_{\hat{n}}$. In practice, the gateway usually has around one second to reply with the new update interval. Further, we discretise the possible update intervals in steps of duration t_S, up to a maximum update interval of T_{max}, an integer multiple of t_S. Discretising time enables us to formulate the problem the gateway faces as follows:

$$\begin{aligned} \underset{T_{\hat{n}}}{maximize} \quad & \min\Big(\mathcal{L}_n(T_n)\Big), \forall\, n = 1, \ldots, N \\ \text{subject to} \quad & \varepsilon_n(t_{\hat{n}} + T_{\hat{n}}) \leq \varepsilon^*, \forall\, n = 1, \ldots, N., \\ & T_{\hat{n}} \in \{t_S, 2t_S, \ldots, T_{max}\}. \end{aligned} \tag{7}$$

At time step $t_{\hat{n}}$, when the gateway receives an observation from sensor $\hat{n}$, the gateway has to select the update interval $T_{\hat{n}}$ that will maximise the lifetime of the sensor (or sensors) in the deployment with the shortest life expectancy. The gateway has to ensure that the accuracy constraint, i.e., $\varepsilon_n \leq \varepsilon^*$, is met for all sensors until the next update is received at time $t_{\hat{n}} + T_{\hat{n}}$. In other words, until the system receives a new update from sensor $\hat{n}$, the value of the estimation error should not exceed the set target on any of the sensors. Additionally, because one sensor's update time potentially affects the lifetime of all other sensors, the gateway might have to select a lower update time for sensor $\hat{n}$. By doing so, the scheduler can prolong the lifetime of another sensor, preferably one with less available energy, to maximise the lifetime, i.e., $\mathcal{L}_n(T_n), \forall n = 1, \ldots, N$.

To solve the problem, the gateway could turn to a numerical solver. However, due to the matrix inversion in Eq. (2), required to determine the MSE value using Eq. (5) for N sensors, the expected computational complexity to solve the proposed problem is of the order of $\mathcal{O}(N^4 \frac{T_{max}}{t_S})$. The required computational power for a conventional numerical solution rises to the power of four with the number of sensors. In practice, the gateway does not have enough time available to determine the sensor's new update interval. Additionally, if conditions in the environment change, e.g., a sensor disconnects, the optimization problem becomes absolute. Everything considered the most practical approach is to employ a DRL algorithm. The DRL removes the constraint of a computationally intense on-line method as all the power is used in off-line training. Consequently, the system can respond in a few milliseconds. The use of Artificial Neural Network (ANN) also allows the system to create a model, free of human influence, for the complex relationships between the energy and correlation. Thus the learned updating strategies are not limited by design choices. Another advantage of DRL is adaptability to environmental conditions changes due to its on-line learning approach.

The system is highly dynamical, as each received observation impacts the covariance model's scaling parameters. As a result, the value of the MSE (Eq. (5)) continuously varies over time. Intuitively, when a sensor has more energy available than others, it should transmit more often, to enable other sensors (which will transmit in the future) to increase their update intervals. Such a problem is ideal for a RL approach because the agent (in our case the gateway) can learn how to take actions that might bring negative reward in the near future but will ultimately increase the long-term reward [36] in prolonging the network lifetime. In the next section, we show how we model the decision making discussed using an RL solution, describing the relevant states, actions, and rewards. Then, by applying a DDPG algorithm, the gateway can arrive at a long-term updating policy to collect accurate observations and prolong the network lifetime.

V. Deep Reinforcement Learning Approach

In DRL, the agent learns its optimal behaviour, i.e., the best long term action to be taken in every state, from interactions with the environment. In our case, the learning agent resides in the gateway. The agent follows a sequence of events. Once a sensor transmits an update, the agent has to respond by setting the sensor's update interval, and by calculating the MSE value it can assess the impact of the set update interval. The agent must also consider the remaining energy available in each battery-powered sensor and their current update intervals when making the decision. To solve the decision-making problem the agent is facing, we employ a DDPG algorithm. Using the DDPG algorithm enables us to design a scheduling mechanism with a high number of possible actions regarding setting the sensor's update interval. There are two main advantages of using a DDPG algorithm over other DRL algorithms: 1) there is a high convergence guarantee even when using non-linear function approximations, e.g., ANN [37] and 2) DDPG is deterministic, meaning that the policy gradient is integrated only over the state space, thus requiring much fewer samples to find the optimal policy in comparison to stochastic algorithms [38]. In our work we follow the DDPG algorithm implementation as presented in [39].

A. Deep Deterministic Policy Gradient Algorithm

DDPG is an actor-critic algorithm and, as the name suggests, consists of two entities/Artificial Neural Network (ANN): the actor taking actions, and a critic that evaluates them. The critic is implemented as a DQN and we denote

its ANN as $Q(\mathbf{s}, a|\theta^Q)$ where θ^Q are weights of the critic's ANN, with a denoting the action the agent takes in state $\mathbf{s}$. Next, we define the actor as a parametric function $\mu(\mathbf{s}|\theta^\mu)$ in which θ^μ represents the actor's ANN weights. In addition, during the training process we initialize the target ANN $Q'(\mathbf{s}, a|\theta^{Q'})$ and $\mu'(\mathbf{s}|\theta^{\mu'})$ with weights $\theta^{Q'}$ and $\theta^{\mu'}$ for critic and actor respectively. The agent selects actions according to its current policy with added noise $a = \mu(\mathbf{s}|\theta^\mu) + \mathcal{N}$, where $\mathcal{N}$ represent added random noise. Note that exploration in a DDPG algorithm is carried out by adding a random value, i.e., noise, to the actor's selected value. Then the agent transitions into a new state $\mathbf{s}'$ and receives reward r. The transition $(\mathbf{s}, a, r, \mathbf{s}')$, also referred to as experience, is stored in memory. When the algorithm is training the ANN it first samples a mini-batch of M experiences from the batch and calculates the target values:

$$h_m = r_m + \gamma Q'(\mathbf{s}'_m, \mu'(\mathbf{s}_m|\theta^{\mu'})|\theta^{Q'}), \tag{8}$$

with m denoting the selected experience from the batch. After determining h_m we can update the critic by minimizing the loss L as:

$$L = \frac{1}{M}\sum_{m=1}^{M}(h_m - Q(s_m, a_m|\theta^Q))^2. \tag{9}$$

With the loss function calculated, the algorithm then updates the actor's policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{M}\sum_{m=1}^{M} \nabla_a Q(\mathbf{s}, a|\theta^Q)|_{\mathbf{s}=\mathbf{s}_m, a=\mu(\mathbf{s}_m)} \nabla_{\theta^\mu}\mu(\mathbf{s}|\theta^\mu)|_{\mathbf{s}_m}. \tag{10}$$

In the last step the DDPG algorithm updates the target ANNs:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau_C)\theta^{Q'}, \tag{11}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau_A)\theta^{\mu'}, \tag{12}$$

where τ_A and τ_C represent the target networks update factor. Note that the role of the target ANNs is to calculate h_m. Using separate target networks along with the replay buffer provide stability during the training process as was established in [40].

Figure 4 illustrates how we adopt DDPG in our scheduling algorithm. In our approach, the gateway performs every computationally demanding task. A low-power sensor only receives a command message that instructs the sensor for how long it should enter sleep mode. By setting the duration of each sensor's sleep mode, the gateway effectively schedules the sensors' transmission of updates. In our mechanism the actor's actions are limited to increasing and decreasing the sleep time, and the critic's role is to evaluate whether the selected change is beneficial to the system. The critic derives a value representing the quality of the selected action using the reward for the actor. The actor then uses the provided quality of action value to adapt its policy accordingly.

Next, we define states, actions, and rewards, i.e., a tuple in $\langle \mathcal{S}, \mathcal{A}, R\rangle$ that enables the gateway to determine each sensor's optimal update interval.

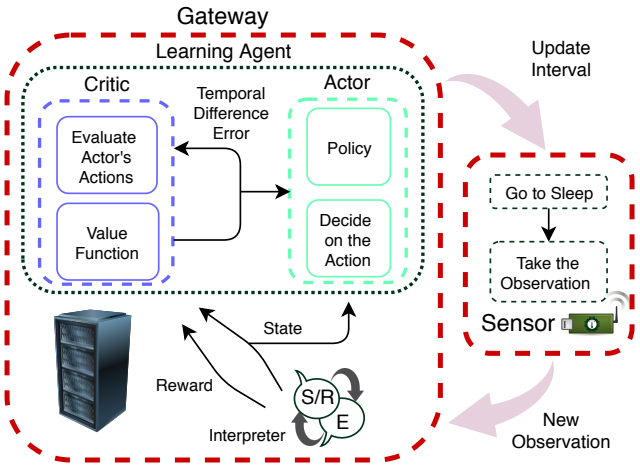

Fig. 4: A high-level overview of the proposed scheduling mechanism implemented with a Deterministic Policy Gradient Algorithm.

B. States, Actions, and Rewards

Upon the n-th sensor's transmission the gateway constructs the state $\mathbf{s}_n \in \mathcal{S}$. The state must capture the critical aspects of the decision-making process in the gateway: sensors update interval, available energy, and the value of the estimation error. To make the decision the gateway has to weight the n-th sensor status (current update time, average estimation error, and energy) against the state of all other sensors. The state $\mathbf{s}_n$ can be expressed as a six-dimensional vector:

$$\begin{aligned}\mathbf{s}_n = (T_n, E_n, \frac{\overline{\varepsilon}_n}{\varepsilon^*}, \left(\prod_{\substack{i=1\\ i\neq n}}^{N} T_i\right)^{\frac{1}{N-1}}, \\ \left(\prod_{\substack{i=1\\ i\neq n}}^{N} E_i\right)^{\frac{1}{N-1}}, \left(\prod_{\substack{i=1\\ i\neq n}}^{N} \frac{\overline{\varepsilon}_i}{\varepsilon^*}\right)^{\frac{1}{N-1}})\end{aligned} \tag{13}$$

with N representing the total number of sensors under the agent's control. The first three dimensions correspond to the transmitting n-th sensor's update interval (T_n), available energy(E_n), and the ratio between MSE and target estimation error ($\overline{\varepsilon}_n/\varepsilon^*$). Relying on the ratio enables the learning agent to perform well even if the target estimation error changes, as we demonstrate in the next section. The last three dimensions of the state vector reveal the state of all other sensors in the system. Using the geometric mean enables us to reduce the number of state inputs, while simultaneously making sure the learning agent captures the most significant information about the environment. For example, the geometric mean provides information to the agent regarding whether the energy level in the majority of the sensors is low or high. Note, that even with a limited number of dimensions for the state input vector, millions of possible different states exist. The gateway uses the interpreter to construct the state vector for the sensor which observation it has just received, and for which it is making the decision.

The learning agent's *action* is limited to either increasing or decreasing the sensors' current update interval. In our implementation, the action value ($a_n \in \mathcal{A}$) returned by the

DDPG algorithm is between -1 and 1, i.e., $\mathcal{A} = [-1, 1]$. To determine the sensors' new update interval, we multiply the received action value by a constant, representing the selected maximum update interval change U_{max}. We calculate the new update interval as follows:

$$T_{\hat{n}} = \min\Big(\max\left(T_n + \lfloor U_{max} a_n \rceil, 1\right), T_{max}\Big), \tag{14}$$

where T_n is the sensors' previous update interval. Note that the value of U_{max} can be relatively large, e.g., hundreds of seconds, and our approach will still perform well. Additionally, the selection of U_{max} dictates the number of possible actions the agent has at is disposal, which is twice the value of U_{max} in time-steps.

We form the *reward* with the learning agent's goals in mind. The learning agent has to ensure that information collection is frequent enough to maintain the freshness of the information and simultaneously try to prolong the sensors' lifetime. We express the reward as:

$$r_n(\bar{\varepsilon}_n, E_n) = \phi r_{acc}(\bar{\varepsilon}_n) + (1-\phi) r_{en}(E_n). \tag{15}$$

$r_{acc}(\bar{\varepsilon}_n)$ is the reward for accurate collection of observations and $r_{en}(E_n)$ is the reward related to the energy aspect of the problem. The weight $\phi \in [0.25, 0.75]$ controls the balance between the reward for accurate collection of observations and the sensors' energy preservation. We restrict the range of the weight ϕ to avoid the reward from being overly weighted towards one goal or the other.

The accuracy reward depends on whether the set average accuracy, ε^*, was satisfied since the n-th sensor's last transmission, as well as on the change in the estimation error since the last transmission. We define the accuracy reward as:

$$r_{acc}(\bar{\varepsilon}_n) = \begin{cases} \text{when} \quad \bar{\varepsilon}_n \leq \varepsilon^* : \\ \quad \left(\frac{\bar{\varepsilon}_n}{\varepsilon^*}\right)^2 + \Upsilon\ \Delta\bar{\varepsilon}_n \\ \text{when} \quad \bar{\varepsilon}_n > \varepsilon^* : \\ \quad \left(\frac{\bar{\varepsilon}_n - \varepsilon^*}{\varepsilon^*}\right)^2 - \Upsilon\ \Delta\bar{\varepsilon}_n \end{cases} \tag{16}$$

where $\Delta\bar{\varepsilon}_n$ represents the change in the average estimation error since the previous transmission. The closer the estimation error is to the target, the greater will be the reward from the first term of the expression. The second term of the accuracy reward steers the learning agent towards keeping the MSE as close as possible to the target ε^*, without exceeding it. The factor Υ is used to balance the contributions of the two parts of the accuracy reward.

Our energy reward exploits the relationship between the update interval and a battery-powered sensor's lifetime. The longer the time between the consecutive updates, the longer the sensors' lifetime will be. Therefore, the selected energy reward is based on how the update interval is increased or decreased, as follows:

$$r_{en}(E_n) = \begin{cases} 1 - \frac{N E_n}{\sum_{i=1}^{N} E_n}, & \text{if} \quad T_{\hat{n}} > T_n \\ 0, & \text{if} \quad T_{\hat{n}} = T_n \\ \frac{N E_n}{\sum_{i=1}^{N} E_n} - 1, & \text{if} \quad T_{\hat{n}} < T_n \end{cases}, \tag{17}$$

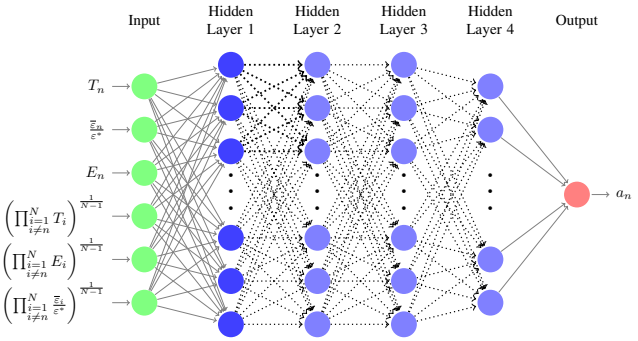

Fig. 5: Actor's ANN structure.

where E_n is the sensor's available energy. If a sensor has above average available energy, the energy reward should encourage the learning agent to make sure that such a sensor updates more often, and vice-versa if a sensor has below average energy.

In the next subsection, we present our implementation of the DDPG algorithm.

C. Artificial Neural Network Structure

To implement the learning agent using DDPG as described in [39] we used Pytorch [41], a standard Python-based library for implementing DRL algorithms. We employ similar ANN structures for the actor and the critic. The actor's ANN consists of an input (state space), output (action value), and four hidden layers with Feedforward Neural Network (FNN) structure, as shown in Figure 5. We use 75 neurons in the first three hidden layers and 25 neurons in the fourth layer. Between each hidden layer, we implemented a 50% dropout layer. We use batch normalization after activation in the first hidden layer. The dropout layers prevent over-fitting, and batch normalization improves the learning speed. We employ the same structure for the critic's ANN, with a slight difference in the activation function used. We use a ReLU activation function for every layer in both ANNs. The only exception is the output layer of the actor's ANN, where we use a Hyperbolic function. Such a difference is required as the actor's output value is limited to values between -1 and 1 while the critic's is not. To train the ANNs we periodically perform batch learning. In each batch training we use 128 experiences. Each experience consists of a state, corresponding action and reward, and the state to which our sensor transits after taking the selected action. Note that 128 experiences are randomly selected from a memory pool of up to 100,000 experiences. While, the exploration in DDPG algorithms is performed by adding noise, i.e., a random value, to the actor's output value. We use the Ornstein-Uhlenbeck process [39] to generate a random value to be added to the selected action value.

In the gateway, we deploy *only one set* of actor's and critic's ANN for the DDPG algorithm to control the update times of all N sensors in the deployment. The main benefit of using only one set of ANNs is the faster collection of necessary experiences for training. Additionally, the system experiences a greater variety in the six-dimensional state vector values, which means that the initial training period is much lower

than if the gateway relied on a separate set of ANNs for each sensor under its control. Using only one set also has a practical benefit as it requires much less computational power at the gateway for the training process.

The **interpreter** has a crucial role in the operation of the proposed scheduler. Its function is to determine the state(the six dimension vector defined in Eq. (13)) and reward (Eq. (15)) whenever the gateway receives an observation from the n-th sensor. To determine both, the interpreter relies on already gathered information regarding the environment stored in the gateway. For example, by examining metadata in the observations, the gateway can learn about the sensors' energy levels. Additionally, these observations are used to extract the covariance model's scaling parameters. The covariance model is then used to determine the estimation error. In other words, the gateway leverages collected observations to acquire knowledge about the environment to make a more informed decision. Such operational logic follows our discussion in the previous section (problem formulation). In our design, the obtained external information is passed to the learning agent trough the interpreter's in the form of the state and reward, as illustrated in the Fig. 4.

We set the start of an episode to when a sensor transmits an observation. When the gateway receives an observation, it uses the interpreter to determine the sensor's state and calculates the reward for the action taken, i.e., the change in the update interval. The sensor state information is then passed to the learning agent to determine the new action; the learning agent calculates the sensor's new update interval and informs the sensor. The sensor will enter sleep mode for the amount of time determined by the learning agent. As soon as the sensor wakes up, the episode ends, and the new episode starts as the sensor transmits a new update.

In the next section, we evaluate the performance of our proposed mechanism using data obtained from a real sensor network deployment.

VI. Evaluation and Results

In this section, we evaluate our proposed scheduling mechanism using observations provided by the Intel Berkeley Research laboratory [26], as well as data collected from multiple sensors deployed in the city of Santander, Spain, as part of the SmartSantander testbed [27]. In our experiments, a simulated sensor transmits an observation with the exact value as was obtained by its real world counterpart. In other words, the data enables us to realistically represent how the observed physical phenomenon, i.e., $Z(\boldsymbol{X}, t)$, varies over time. We evaluate our mechanism using five different datasets: two from Intel (temperature and humidity), and three from SmartSantander (temperature, humidity, and ambient noise). We split the evaluation into three parts: In the first part, we demonstrate that the proposed scheduling mechanism can learn the optimal behaviour and that using our approach can significantly extend the sensors' lifespan. In the second part of our evaluation, we perform complexity and run-time analysis. Finally in our third part, we highlight the energy-aware aspects of our proposed scheduling mechanism.

In our simulation, each sensor is assigned observations obtained by a real sensor and location as it had in real deployment. Meaning, that when a n-th sensor decides to transmit an observation ($\boldsymbol{X}_n$) at time t_n, the gateway receives a measurement, e.g., temperature reading, which the simulated sensor real counterpart has obtained in the corresponding time moment. The Intel laboratory dataset provides observations collected over nine days of measurements from 50 sensors. The observations are collected very frequently (every 31 seconds), providing us with a ground truth of observed values for the evaluation process. On the other hand, the Smart Santander data represents a realistic deployment of sensors in a smart city environment. In our analysis, we use data obtained in the first nineteen days of April 2018. We rely on data from twenty temperature, ten humidity, and eight ambient noise sensors. In contrast to the Intel data, every sensor transmitted observations at different time intervals, and sometimes there were a few hours during which a sensor did not send any observation. We use the Amelia II software program [42], a tool to generate missing data, to produce the missing values. To generate these values, we used observations from the sensor that collected the highest number of observations; as a consequence, we had to remove that sensor from each SmartSantander dataset, to avoid adding bias in the evaluation.

Due to our use of real observations to model the environment, we had to insert an extra step at the evaluation stage. The learning agent could recognise patterns in the collected observations and adapt its behaviour accordingly. In other words, the agent could overfit its policy to the dataset. To avoid such a scenario, we have to ensure that the environment the agent interacts with during evaluation is new. To that end, we had to split each dataset into two parts. The first part is used for exploration, during which the agent learns the policy. The second part is used for the evaluation of its performance. We split the Intel dataset into six days used for exploration and three days for evaluation, while for SmartSantander we use the last six days for evaluation and the rest to train the agent.

TABLE I: Static Simulation Parameters

Parameter	Value	Parameter	Value
U_{max}	$250s$	P_c	$15uW$
E_0	$6696J$	E_{tr}	$78.7mJ$
T_{start}	$900s$	Υ	10
ε^*	0.01	ϕ	0.5
T_{max}	$7200s$	t_S	$10s$

We list system parameters that are kept constant throughout our evaluation process in Table I. We selected static simulation energy parameters by assuming that each of the sensors is powered by a single non-rechargeable Lithium coin battery with a capacity of 620 mAh, which provides us with the value for E_0. The selected energy consumption parameters, i.e., P_c and E_{tr}, mimic the power consumption of an IoT sensor using a LoRaWAN radio. We obtain the power parameters following the analysis presented in [43]. The selected U_{max} yielded the best average performance at the end of the training phase for all five datasets. Altogether our agent can select among 51

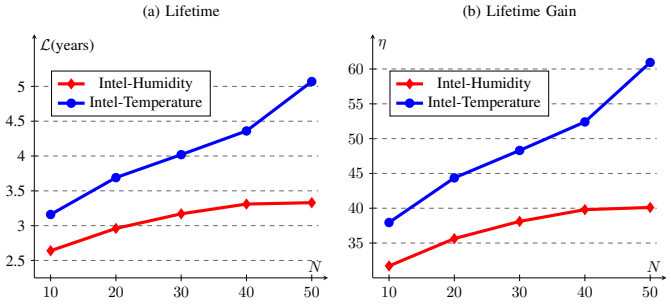

Fig. 6: Lifetime of sensors and lifetime gain achieved by our updating mechanism as the number of sensors under its control increases.

different actions, i.e., changes of update interval.

T_{start} represents a suitable starting update time value, while T_{max} represents the maximal value that should be allowed between two consecutive updates from one sensor. At the start of every simulation, we set the same initial update time for every sensor. ϕ and ε^* are set to the value stated in the Table unless they are parameters that we change in the presented experiment. We perform simulations in ten-second time-steps. In Table II we list the DRL solution hyperparameters we determined through a grid search to be most suitable. Note that we multiply the calculated rewards by a factor of 10 to improve the training process, as higher reward values tend to reduce the time required for the DDPG algorithm to arrive at the optimal policy [44].

TABLE II: Hyperparameters

	Hyperparameter	Value
DQN	Learning rate	0.5
	Discount factor	0.2
	Explore rate	0.15
	Target ANN soft update	10^{-3}
	ANN Learning rate	10^{-3}
	Batch size	32
	Memory size	2×10^4
	Optimizer	Adam
	Loss Function	MSE
DDPG	Actor's ANN learning rate τ_A	10^{-4}
	Critic's ANN learning rate τ_C	10^{-4}
	Target ANN soft update	10^{-3}
	Critic's Discount factor	0.99
	Batch size M	128
	Memory size	10^5
	Optimizer	Adam
	Loss Function	MSE

A. Performance Evaluation

In this subsection, we evaluate our scheduling mechanism's ability to maintain accurate observations and compare it to other scheduling approaches.

First, we test the updating mechanism performance as the number of sensors, N, under its management increases. The Intel datasets offer us a maximum of 50 sensors, and for cases in which $N < 50$, we randomly selected a subset of these sensors and then repeated the experiment several times. As expected, increasing the number of sensors leads to more correlated information available, and therefore, the gain of using correlated information increases with the number of sensors. The benefit of using correlated information is higher when observing temperature, due to higher correlation exhibited in the observations collected. We calculate the expected lifetime using Eq. (6). In Fig. 6(a) we use both Intel datasets, humidity and temperature, and calculate the expected lifetime using the average update interval that sensors achieved in the experiment. Additionally, to demonstrate the performance improvement brought by our solution, we calculate the lifetime gain η, plotted in Fig. 6(b). We define this gain as the ratio between the lifetime achieved using our mechanism and that achieved in the original datasets. The resulting high gains arrive from the originally selected update intervals adopted in the datasets, which would lead sensors to last only a month (a time duration that coincides with the original time the sensors in the Intel lab were deployed). By adopting our approach, the same sensor deployment could last for five years, as the average update interval our DRL solution has determined is 49 minutes.

In Fig. 7 we compare the performance of our DDPG-based scheduling mechanism, in terms of achieved sensor lifetime, to three different methods of obtaining update times:

1) **Original**: as a baseline, we take the update times adopted in the Intel Lab and Smart Santander datasets. The baseline case reveals the original expected lifetime of deployments, if the sensors were battery-powered. Note that for the SmartSantander datasets, we use the update time of the sensor that collected the most observations.
2) **DQN**: update times we obtain by using the Deep Q-Network approach we proposed in [25]. Note that the reward function we use for the DQN approach is the same as we defined in Eq. (15), while the action space consists of five actions. The scheduling mechanism can take an action to increase the sensor's current update interval by one or ten time-steps, decrease it for one or ten time-steps, or keep it as it was.
3) **Ideal Scheduler**: in the ideal case, the network controller acts essentially as an oracle that knows the ground truth and can obtain observations from any sensor on demand. With the ideal scheduler, sensors are asked to transmit only when the average error of estimated values (of temperature, humidity, ambient noise) exceeds the set threshold ε^*. The lifetime the ideal scheduler obtains is the maximal possible lifetime the sensors can achieve while still maintaining the set accuracy constraint. In other words, the ideal scheduler embodies the optimal solution.

In Fig. 7, we show the expected lifetime we achieved using all four approaches for all five datasets. We calculated the lifetime using Eq. (6) and using values of average update times that we obtained with each approach. The results show that our mechanism is capable of finding update intervals that are very close to the ideal one. The only exception is for the case of Intel lab temperature data. In that case, a higher U_{max} would enable the agent to get results closer to the ideal case. However, by keeping the U_{max} constant throughout our validation we can demonstrate the system's robustness to

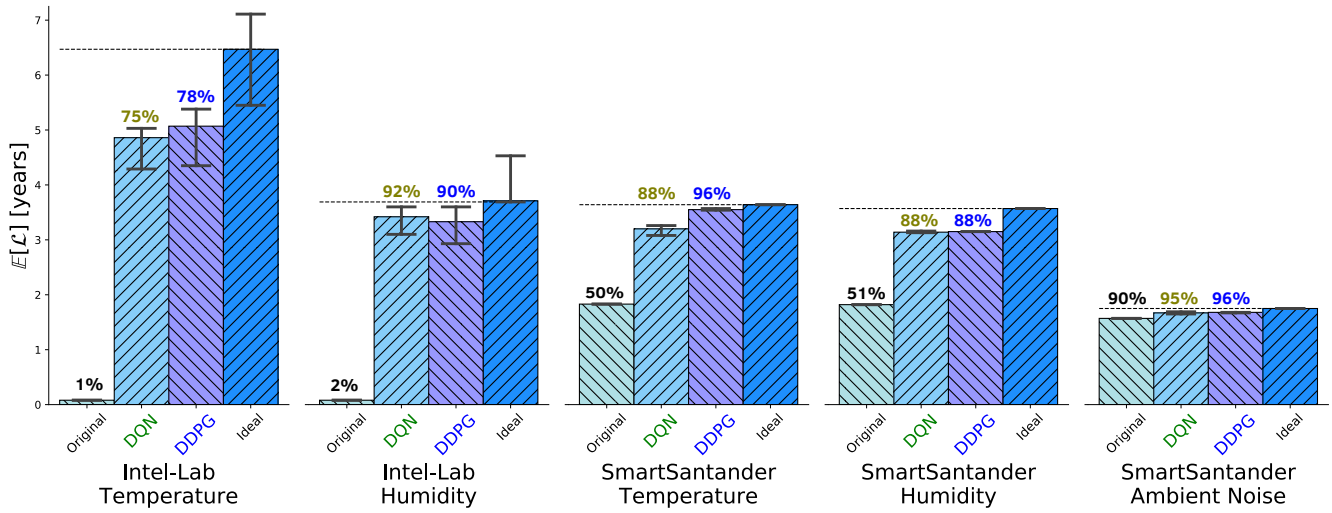

Fig. 7: Achieved expected lifetime in years with every approach for all five datasets. The number on top of each bar plot reveals how close each approach is to the Ideal case.

multiple different scenarios. Similarly, for the DQN solution, we would have to increase the number of actions to be able to improve the overall performance. However, with a higher number of actions, the required training time increases drastically, which limits the DQN solution's applicability to the proposed problem. Table III shows that the DDPG approach achieves $91 - 92\%$ of the error threshold, for every dataset, while the DQN approach varies slightly more due to the mechanism's inability to adapt to the environmental changes promptly.

The importance of faster adaptability to an ever-changing environment is paramount in real deployments. In this criterion, the DDPG solution greatly outperforms the DQN approach even if results in Fig. 7 indicate very similar performance in terms of achieved lifetime. Additionally, we noticed that the DQN approach requires a longer exploration time, e.g., for the Intel dataset, we iterated over the exploration part of the data three times, in comparison to two iterations required to train the system using DDPG.

To improve the performance of the proposed DDPG approach in comparison to the ideal, the apparent solution is to provide the agent with more information, i.e., expand the ANN input state space. For example, adding information regarding the average distance to neighbouring sensors would help. However, adding the average distance as part of the state information would decrease the generality of our solution. Intuitively, providing the agent with direct information regarding all sensors in the system (current update interval, average estimation error, and energy level) should lead to better results. Unfortunately, a significantly larger state space would lead to longer computational time and even possibly a degradation in performance, as an agent could have a hard time differentiating between the more and the less relevant information, e.g., which sensor's updates are most relevant for the current decision.

The minimum and maximum expected lifetimes among all sensors are shown as a solid line in the middle of each bar

TABLE III: Obtained $\bar{\varepsilon}$ per dataset.

Dataset	ε^*	DQN	DDPG	Ideal
Intel-Tem	0.01	0.0088	0.0092	0.01
Intel-Hum	0.01	0.0093	0.0091	0.01
SmartSan-Tem	0.01	0.0091	0.0091	0.01
SmartSan-Hum	0.01	0.0094	0.0091	0.01
SmartSan-Amb	0.01	0.0098	0.0092	0.01

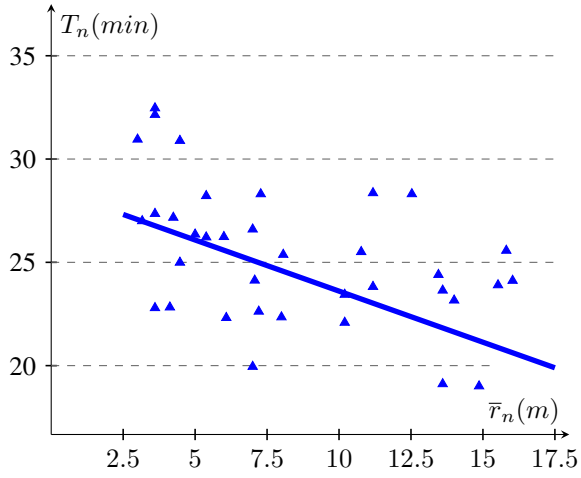

Fig. 8: Sensor's update interval change depending on the average distance to other sensors in the network.

in Fig. 7. The variation is especially noticeable in the case of Intel-Lab data, indicating that some sensors benefit more from using correlated information than others. For this dataset, sensors are deployed on the same floor, and those in the middle of the room are able to benefit more from correlated measurements collected from other nearby sensors.

We remind the reader that distances are not part of the information explicitly furnished to the learning agent. As we show in Fig. 8, in which we plot the achieved average update interval as a function of the average distance to other sensors, some

sensors benefit more than others from information collected by their neighbors. Note that in our system each sensor represents the location at which we observe the physical phenomenon. More specifically, it appears that the average distance to other sensors is inversely proportional to the achieved update interval. We obtained the result using temperature test data for the Intel-lab dataset. The line in Fig. 8 is obtained using linear regression. Such behaviour is interesting as the agent does not have direct information regarding the distances, yet it can infer it from other available information. However, these differences appear only in the short term, as the energy-awareness ability of our proposed scheduling mechanism will in the longer run prevent one sensor from updating much more often than others and avoid the depletion of one sensor's battery much earlier than the others.

B. Complexity and Run-time analysis

The computational complexity of our DRL solution depends solely on the dimension of the action space, D_a, the dimension of the state space, D_s, number of layers L, and the number of neurons in each hidden layer, W. Each invocation of forward propagation, i.e., when the agent makes its decision, is linear with respect to the ANN parameters. This means that we can write the number of computations required for the actor as $\mathcal{O}(D_s LW)$, and, similarly, as $\mathcal{O}((D_a + D_s)LW)$ for the critic. Considering that we employ ANNs with only four hidden layers, i.e., L=4, and with 75 neurons (25 in the fourth hidden layer) per layer, the resulting number of computations is relatively small. Our agent can respond within 2 to 3 ms, thus satisfying the timing constraints set by the sensors' communications technology. For comparison, the DQN solution as implemented consist of only one ANN, with an input dimension D_{sQ} (state-space), number of layers, L_Q, number of neurons in hidden layers, W_Q, and actions space D_{aQ}. Note that the DQN implementation in [25], used in our comparison, uses a ANN with two hidden layers, each with 24 neurons.

In Table IV, we list computations required for both DRL solutions and on average response time. DDPG requires more time due to the higher number of computation. However, the computation time is well within the time the gateway has at is disposal to respond. For comparisons, the same system would need 7-12 seconds for $N = 10$ to numerically resolve the optimisation problem. Furthermore, while for a numerical solution to the optimisation problem, the complexity rises with the number of sensors to the power of four, in our problem.

TABLE IV: Comparison of Complexity, Implementation Run-time

	Computational Complexity	Average response time
DQN	$\mathcal{O}(D_{sQ}L_Q W_Q)$	$\sim 2\ ms$
DDPG	$\mathcal{O}(D_s LW)$ + $\mathcal{O}((D_a + D_s)LW)$	2-3 ms

The goal of each DRL solution is to find an updating strategy capable of prolonging the lifetime of battery-powered sensors. Experimenting with the Intel Lab dataset, we estimate that a DDPG based scheduler achieves 80% performance of the fully converged solution. After the initial fast learning period,

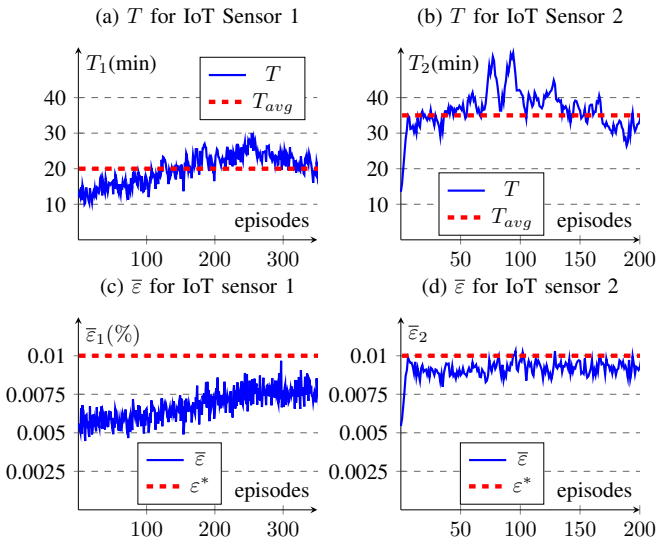

Fig. 9: Two IoT sensors with different battery levels learning over a number of episodes. Sensor 1's battery level is at 75% while sensor 2 has 25% of its battery life remaining.

the performance slowly improves over the next few days due to on-line learning. In comparison, a DQN-based scheduler requires almost three days to achieve a similar performance a DDPG-based solution learns in one day. Furthermore, the DQN solution requires almost twice as many days to converge fully. Such behaviour indicates that the DDPG implementation adapts faster to a changing environment.

C. Energy-aware Scheduling

In this subsection, we demonstrate the energy-aware capabilities of the proposed mechanism. We evaluate the energy-aware performance using Intel-laboratory temperature data, and in all experiments the scheduling mechanism controls 50 sensors.

First, we show the changes in the update interval and $\bar{\varepsilon}_n$ over a number of episodes for two sensors. In Fig. 9(a) we plot the update interval over a number of episodes for a sensor with above-average available energy (75% of remaining battery life), while in Fig. 9(b) we plot update intervals of sensors with below-average energy (25% of battery life). Note that we set the energy levels for all other sensors to 50%, and that the two sensors were randomly selected. As we show, our updating mechanism sets the update interval of a sensor with less remaining energy significantly higher in comparison to the update interval of a sensor with more energy available. By setting different update intervals, the mechanism can, in the longer run, balance the energy levels among sensors. In other words, the mechanism forces the sensor with more energy to update more often than it would otherwise need to, in order to help preserve the energy of the sensor with shorter battery life. As we show in Fig. 9 (c) and (d), the set error threshold is not exceeded.

Fig. 10 shows the ratios of update rates of sensors with different energy levels. We set the energy level of 90 percent of sensors (45 out of 50) in the dataset to a fixed value. Then we change the energy level of the remaining ten percent of

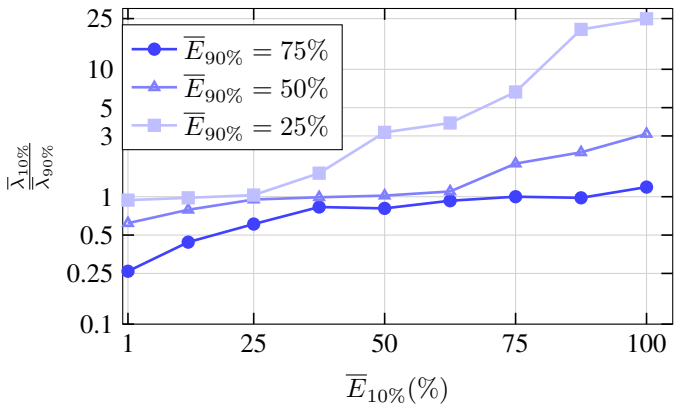

Fig. 10: The change in update rate ratio as the percentage of the battery in sensors change.

sensors (5) in steps from 1% to 100% of remaining battery life. In each step we iterate over the test data and report the ratio between the achieved average update rate of the two groups, i.e., $\frac{\overline{\lambda}_{10\%}}{\overline{\lambda}_{90\%}}$. As expected, when sensors have the same energy level, they will transmit with roughly the same update rate. We can observe an intriguing behaviour when sensors are close to depleting their energy. When a few sensors have much more energy than others, they will transmit new observations much more often, even 25 times more often when they have a full battery and the other sensors' energy level is at 25%. By doing so, the mechanism effectively decides to use the energy of a few sensors to prolong the lifetime of all others. Over time, the tendency will be towards achieving more uniform energy use throughout the network.

VII. Conclusion

In this paper, we have proposed a DRL-based energy-aware scheduling mechanism capable of significantly prolonging the lifetime of a network of battery-powered sensors without hindering the overall performance of the sensing process. We have demonstrated, using real-world observations, that the performance of our proposed mechanism is near-optimal. Additionally, the proposed mechanism is capable of setting update intervals depending on the energy available on devices. Such behavior ensures that battery-powered sensors deployed at the same time will also expire at the same time, thus enabling infrastructure providers to replace the entire sensor deployment simultaneously. As such, the energy-aware scheduling mechanism we proposed in this paper can have a profound impact on the suitability of the future deployments of IoT sensing devices.

In our future work, we will focus on a network of sensors using diverse primary power sources, e.g., mains powered or event-based energy harvesting. In such a case, the resulting scheduling policy will depend on the type of power source the low-power sensor relies on.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for Smart Cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans. Ind. Informat.*, vol. 10, no. 4, pp. 2233–2243, Jan. 2014.

[3] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big Data in Smart Farming A review," *Agricultural Systems*, vol. 153, pp. 69–80, May. 2017.

[4] J. Hribar, M. Costa, N. Kaminski, and L. A. DaSilva, "Using Correlated Information to Extend Device Lifetime," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2439–2448, Apr. 2019.

[5] A. Kosta, N. Pappas, V. Angelakis *et al.*, "Age of Information: A New Concept, Metric, and Tool," *Found. Trends Netw.*, vol. 12, no. 3, pp. 162–259, Nov. 2017.

[6] S. Kaul, R. Yates, and M. Gruteser, "On Piggybacking in Vehicular Networks," in *Proc. IEEE GLOBECOM*. Houston, TX, USA, Dec. 2011, pp. 1–5.

[7] R. D. Yates, "Lazy is Timely: Status Updates by an Energy Harvesting Source," in *Proc. ISIT*. Hong Kong, Jun. 2015, pp. 3008–3012.

[8] A. E. Kalor and P. Popovski, "Minimizing the Age of Information from Sensors with Common Observations," *IEEE Wireless Commun. Letters*, Oct. 2019.

[9] Z. Jiang and S. Zhou, "Status from a Random Field: How Densely Should One Update?" in *Proc. ISIT*. Paris, France, Jul. 2019, pp. 1037–1041.

[10] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad hoc networks*, vol. 7, no. 3, pp. 537–568, May 2009.

[11] T. Rault, A. Bouabdallah, and Y. Challal, "Energy efficiency in wireless sensor networks: A top-down survey," *Computer Networks*, vol. 67, pp. 104–122, Jul. 2014.

[12] L. A. Villas, A. Boukerche, H. A. De Oliveira, R. B. De Araujo, and A. A. Loureiro, "A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks," *Ad Hoc Networks*, vol. 12, pp. 69–85, Jan. 2014.

[13] H. Yetgin, K. T. K. Cheung, M. El-Hajjar, and L. H. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 828–854, 2nd Quart., 2017.

[14] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque, "Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 181–194, 1st Quart., 2014.

[15] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y.-C. Liang, and D. I. Kim, "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.

[16] F. Li, K.-Y. Lam, Z. Sheng, X. Zhang, K. Zhao, and L. Wang, "Q-Learning-Based Dynamic Spectrum Access in Cognitive Industrial Internet of Things," *Mobile Networks Appl.*, vol. 23, no. 6, pp. 1636–1644, Dec. 2018.

[17] M. Mohammadi, A. Al-Fuqaha, M. Guizani, and J.-S. Oh, "Semisupervised Deep Reinforcement Learning in Support of IoT and Smart City Services," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 624–635, April 2018.

[18] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan, "Machine learning in wireless sensor networks: Algorithms, strategies, and applications," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1996–2018, 4th Quart. 2014.

[19] J. Zheng, Y. Cai, X. Shen, Z. Zheng, and W. Yang, "Green Energy Optimization in Energy Harvesting Wireless Sensor Networks," *IEEE Commun. Mag.*, vol. 53, no. 11, pp. 150–157, Nov. 2015.

[20] F. A. Aoudia, M. Gautier, and O. Berder, "RLMan: An Energy Manager Based on Reinforcement Learning for Energy Harvesting Wireless Sensor Networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 408–417, Jun. 2018.

[21] J. Du, H. Chen, and W. Zhang, "A deep learning method for data recovery in sensor networks using effective spatio-temporal correlation data," *Sensor Review*, no. 2, pp. 208–217, Mar. 2019.

[22] 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 Things J.*, vol. 5, no. 4, pp. 2375–2385, Aug. 2018.

[23] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. Rodrigues, X. Kong, J. Huang, and R. Y. Kwok, "Deep Reinforcement Learning for Intelligent Internet of Vehicles: An Energy-Efficient Computational Offloading Scheme," *IEEE Trans. Cog. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[24] M. K. Sharma, A. Zappone, M. Assaad, M. Debbah, and S. Vassilaras, "Distributed Power Control for Large Energy Harvesting Networks: A Multi-Agent Deep Reinforcement Learning Approach," *IEEE Trans. Cog. Commun. Netw.*, vol. 5, no. 4, pp. 1140–1154, Dec. 2019.

[25] J. Hribar, A. Marinescu, G. A. Ropokis, and L. A. DaSilva, "Using Deep Q-learning To Prolong the Lifetime of Correlated Internet of Things Devices," in *Proc. IEEE ICC Workshops*. Shanghai, China, May 2019, pp. 1–6.

[26] P. Bodik, W. Hong, C. Guestrin, S. Madden, M. Paskin, and R. Thibaux, "Intel Lab Data," *Online dataset*, Mar. 2004. [Online]. Available: http://db.csail.mit.edu/labdata/labdata.htm

[27] V. Gutiérrez, E. Theodoridis, G. Mylonas, F. Shi, U. Adeel, L. Diez, D. Amaxilatis, J. Choque, G. Camprodom, J. McCann *et al.*, "Co-Creating the Cities of the Future," *Sensors*, vol. 16, no. 11, pp. 1971–1997, Nov. 2016.

[28] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Consensus in Ad Hoc WSNs With Noisy LinksPart II: Distributed Estimation and Smoothing of Random Signals," *IEEE Trans. Signal Proc.*, vol. 56, no. 4, pp. 1650–1666, Apr. 2008.

[29] A. V. Oppenheim and G. C. Verghese, *Signals, Systems and Inference*. Pearson, Mar. 2015.

[30] N. Cressie and H.-C. Huang, "Classes of Nonseparable, Spatio-Temporal Stationary Covariance Functions," *J. Amer. Statis. Assoc.*, vol. 94, no. 448, pp. 1330–1339, Dec. 1999.

[31] T. Gneiting, "Nonseparable, Stationary Covariance Functions for Space-Time Data," *J. Amer. Statis. Assoc.*, vol. 97, no. 458, pp. 590–600, Jun. 2002.

[32] Y. Chen and Q. Zhao, "On the Lifetime of Wireless Sensor Networks," *Communications letters*, vol. 9, no. 11, pp. 976–978, Nov 2005.

[33] C. Bormann, M. Ersue, and A. Keranen, "Terminology For Constrained-node Networks," Internet Engineering Task Force, Tech. Rep., May 2014.

[34] G. Tsoukaneri, M. Condoluci, T. Mahmoodi, M. Dohler, and M. K. Marina, "Group Communications in Narrowband-IoT: Architecture, Procedures, and Evaluation," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1539–1549, Jun. 2018.

[35] *SX1272/73 - 860 Mhz to 1020 MHz Low Power Long Range Transceiver Datasheet*, SAMTECH Corporation, Jan. 2019, revision 4. [Online]. Available: https://www.semtech.com

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, Oct. 2018.

[37] S. Bhatnagar, M. Ghavamzadeh, M. Lee, and R. S. Sutton, "Incremental Natural Actor-critic Algorithms," in *Proc. NIPS*. Vancouver, Canada, 2008, pp. 105–112.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *Proc. ICML*. Beijing, China, Jun. 2014, pp. 1–9.

[39] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous Control With Deep Reinforcement Learning," *arXiv preprint arXiv:1509.02971*, Sep. 2015.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari With Deep Reinforcement Learning," *arXiv preprint arXiv:1312.5602*, Dec. 2013.

[41] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *Proc. NIPS-Workshop*. Long Beach, CA, USA, Dec. 2017, pp. 1–4.

[42] J. Honaker, G. King, and M. Blackwell, "Amelia II: A Program for Missing Data," *J. Statistical Software*, vol. 45, no. 7, pp. 1–47, Dec. 2011.

[43] M. Costa, T. Farrell, and L. Doyle, "On Energy Efficiency and Lifetime in Low Power Wide Area Network for The Internet of Things," in *Proc. IEEE CSCN*. Helsinki, Finland, Sep. 2017, pp. 258–263.

[44] J. Sorg, R. L. Lewis, and S. P. Singh, "Reward Design Via Online Gradient Ascent," in *Proc. NIPS*. Vancouver, Canada, Dec. 2010, pp. 2190–2198.