Timely and sustainable: Utilising correlation in status updates of battery-powered and energy-harvesting sensors using Deep Reinforcement Learning

Jernej Hribar a,∗, Luiz A. DaSilva b, Sheng Zhou c, Zhiyuan Jiang d, Ivana Dusparic a

a CONNECT, Trinity College Dublin, College Green, Dublin 2, Dublin, Ireland
b Commonwealth Cyber Initiative, Virginia Tech, Arlington 22203, VA, USA
c Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China
d Shanghai Institute for Advanced Communication and Data Science, Shanghai University, Shanghai 200444, China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Age of Information
Energy harvesting
Internet of Things
Energy efficiency
Deep Deterministic Policy Gradient

ABSTRACT

In a system with energy-constrained sensors, each transmitted observation comes at a price. The price is the energy the sensor expends to obtain and send a new measurement. The system has to ensure that sensors’ updates are timely, i.e., their updates represent the observed phenomenon accurately, enabling services to make informed decisions based on the information provided. If there are multiple sensors observing the same physical phenomenon, it is likely that their measurements are correlated in time and space. To take advantage of this correlation to reduce the energy use of sensors, in this paper we consider a system in which a gateway sets the intervals at which each sensor broadcasts its readings. We consider the presence of battery-powered sensors as well as sensors that rely on Energy Harvesting (EH) to replenish their energy. We propose a Deep Reinforcement Learning (DRL)-based scheduling mechanism that learns the appropriate update interval for each sensor, by considering the timeliness of the information collected measured through the Age of Information (AoI) metric, the spatial and temporal correlation between readings, and the energy capabilities of each sensor. We show that our proposed scheduler can achieve near-optimal performance in terms of the expected network lifetime.

1. Introduction

Age of Information (AoI) is a rapidly expanding research topic in Information Theory [1–5]. The AoI metric captures the timeliness of collected information in the system, measured as the time elapsed from when a source of information, e.g., a sensor, has generated the latest status update received at the intended destination, e.g., a gateway. Each status update contains information the sensor has obtained, e.g., a measurement. Status updates are time-stamped, which reveals when the sensor took the measurement. The AoI metric is applicable to a variety of Internet of Things (IoT) scenarios [6], as it can be used to determine how often a sensor should transmit its information [7–9]. Identifying the optimal updating strategy greatly improves the energy efficiency of IoT sensors and extends the lifetime of the network.

In a system with energy-constrained sensors, the energy cost of each transmitted status update needs to be considered. The system has to ensure that sensors’ updates are timely and capture the observed phenomenon accurately. For example, in a smart farming scenario, humidity sensors placed in the ground have to report frequently enough to allow the service overseeing the irrigation system to water the fields in time to achieve the desired soil conditions for the crops. In a large-scale system, multiple sensors often collect observations which are correlated in time and space, meaning that an observation from one sensor reveals some information about the current state of the observed phenomenon at other sensors’ locations. In such a system there is a fundamental trade-off between the sensors’ available energy, their updating strategy, and how accurate and up-to-date the information obtained at the gateway is.

The inexpensive IoT sensors typically used in large-scale deployments are heavily constrained in terms of processing power and memory [10], which limits their ability to optimise their updating strategy. Additionally, when making the decision of how often to transmit updates, we have to consider that the radio on these information sensors typically relies on a wake-up scheme [11] to preserve energy. This means that the other network entities, e.g., a gateway, can only communicate with the sensor while it is awake. In other words, the gateway cannot request a status update on demand. For example, if the sensor relies on a Long Range Wide Area Network (LoRaWAN) radio operating in class A, the gateway has only two short time-windows of opportunity to send a command to a sensor after it has transmitted a
status update. Such a mode of operation is common to other Low-Power Wide-Area Network (LPWAN) technologies as well.

In the past, a few works, including our own, have explored the impact of correlation on the Age of Information (AoI) [9,12–18]. The works of [13–16] focused on minimising the value of AoI by taking advantage of correlated information, while the work in [9] and [12] established a connection between the estimation error and the AoI. The estimation error considers that information regarding a phenomenon at a given location can be estimated from recent readings at nearby sensor locations, and thus reveals the value of outdated information to the system. By using the estimation error it is possible to adopt an update strategy that can prolong the lifetime of battery-powered sensors while ensuring the timeliness of information [17,18]. In other words, by reducing the number of status updates the sensors transmit, we can make IoT deployments more sustainable.

The system has to consider the state of all sensors when setting the updating strategy. The available energy at the sensor, the battery capacity, and the ability to employ EH dictate how often sensors can (or should) transmit a status update. In our system, we consider sensors powered by a non-rechargeable battery or by EH. For a single information sensor relying on EH, it is possible to derive the optimal updating strategy [4,5]. Similarly, the author in [8] derived a performance bound for the average AoI that a system of multiple EH sensors can achieve. Unfortunately, a variety of external, often unpredictable, factors impact the rate at which EH collects energy; these include the weather, the intensity of vibrations or light, temperature, etc. Furthermore, two sensors may rely on the same EH technique, e.g., a solar panel, but their energy arrival rate may still not be the same due to external factors, e.g., one sensor sits in the shade. Additionally, due to correlation in the information that sensors collect, an update from one sensor has an impact on when additional readings are needed from other sensors in the system. In such a system, the updating strategy that each sensor must adopt to transmit periodic updates should adapt to the ever-changing network conditions. In this paper, we propose the use of Deep Reinforcement Learning (DRL) to dynamically determine the sensors’ updating strategy by considering their available energy, their current update interval, and the accuracy of the observations collected by the system.

The main contributions of this paper are as follows:

1. We propose and describe the problem of optimising the lifetime of battery-powered sensors and up-time of EH-powered sensors while keeping the accuracy of collected observations under a set threshold.
2. We design a Deep Deterministic Policy Gradient (DDPG)-based scheduling mechanism capable of learning and dynamically adjusting the sensors’ update interval in such a way to ensure timely and accurate collection of correlated status updates. The unique advantage of our proposed solution is a scheduling policy that is capable of prolonging the lifetime of battery-powered sensors by taking advantage of energy collected by EH-powered sensors: we demonstrate such gains to be over 15%.

The remainder of this paper is structured as follows. In the next section, we present the most relevant related work. In Section 3 we describe the system model, explain how we use the estimation error to determine the timeliness of information in a system of correlated information sources, and describe how an EH-powered sensor collects energy. This is followed by Section 4, in which we describe the problem of scheduling updates of correlated sources to achieve a set accuracy level while preserving available energy. We present the proposed scheduler for sensors’ updates in Section 5, followed by an assessment of the scheduler’s performance in Section 6. Finally, we offer concluding remarks in Section 7.

2. Related work

In this work we propose a scheduling mechanism to extend the lifetime of an IoT deployment consisting of some sensors that are battery-constrained and some that employ EH, all of which collecting information that exhibits spatial and temporal correlation. Our scheduler employs DRL to arrive at effective and adaptive scheduling decisions. To our knowledge, there are no other works that consider freshness of information and energy-aware scheduling for a system with a mix of sensors powered by non-rechargeable batteries and by an EH system with the goal of prolonging battery-powered sensors’ lifetime. We discusses related work in three areas: AoI, Wireless Sensor Networks (WSNs), and DRL.

Age of Information (AoI): As we mentioned in the introduction, correlation in status updates was explored in only a few works, including our own [9,12–18]. Moreover, this paper is the only one that also considers sensors being powered by EH. Perhaps the closest related work among the ones listed above is our own work in [17]. In [17,18] we proposed an energy-aware scheduler for correlated battery-powered sensors. However, those papers do not consider sensors powered by EH. The dynamic energy collection of such sensors led us to alter the prior solution substantially. Other works focused on one time-correlated source [13], or considered a static environment [9,12].

Some AoI related works have analysed a system of energy-harvesting sources, but none of those works considered correlation. For example, in [1] a single source using EH is considered. The author established that a source of information powered by EH should not transmit a status update right after it has collected enough energy. A more in-depth analysis of the behaviour of a system of sources powered by EH was carried out by Zheng et al. [19]. The authors show that the update intervals have to be meticulously selected to optimise the timeliness of status updates. Both papers point to the fact that selecting an updating strategy for a system with EH is a non-trivial task.

Wireless Sensor Networks (WSNs): An initial analysis on how to leverage correlation to improve the sensors’ energy efficiency was carried out by Vuran et al. [20]. Works that followed mostly took advantage of correlation by predicting values to be collected by sensors, thus proposing ways to reduce overall transmissions. In many cases, those works consider complex network topologies, and most of the energy savings from the scheduling schemes proposed derive from more efficient routing [21,22]. In contrast, our work does not rely on information prediction, and our proposed scheduling mechanism operates in a network with star topology, as is often the case in sensor networks, with one gateway collecting information from multiple sensors. The current trends in IoT deployment strongly indicate the LPWAN communications technology will prevail in future deployments [23]. Even fewer works have taken into account correlation in systems where sensors are powered by EH [24,25]. The work in [24] focused on designing optimal scheduling policies for EH devices by considering time-correlation in energy arrival. In [25], the authors propose a Markov Decision Process (MDP) scheme for sensors to decide when they should transmit. The environmental map, i.e., the value of the observed physical process, was then reconstructed from collected observations at the gateway by exploiting correlation. However, in contrast to our approach, theirs is not adaptable to changes and it requires additional energy from sensors when updating the MDP model residing on them.

Deep Reinforcement Learning (DRL): Reinforcement Learning (RL) and DRL have been applied to manage a system of sensors powered by EH [26–31]. For example, in [26] and [27], the authors used RL to design a power management solution capable of preventing outages, an outage being defined as when an EH sensor completely depletes its energy. This is a feature that our scheduler also has. The authors in [28] designed a power control mechanism capable of maximising the throughput of EH devices. To build such a solution, they used a multi-agent approach, where each agent is a source implemented with a Deep Q-Network (DQN) algorithm. Similarly, a transmission policy using a DQN algorithm is also investigated in [29]. In their work, the main objective is to find a policy to adjust transmission power and modulation level optimally. In [30] the authors propose a scheme
capable of scheduling the transmissions based on the predicted energy of EH sensors. In [31], the authors employ a DDPG algorithm to develop an energy management solution for a system of EH-powered devices. In contrast to the aforementioned work, we consider a single system of sensors collecting correlated information. Our proposed DDPG-based scheduler takes advantage of available correlated information obtained by EH-powered sensors to prolong the lifetime of battery-powered sensors without impacting the system’s performance.

3. System model

We consider a system consisting of \( N \) sensors, i.e., information sources, powered by a non-rechargeable battery or an EH system. Sensors are observing a physical phenomenon \( Z(x, t) \) distributed in space \( x \) and evolving in time \( t \) as depicted in Fig. 1. The main role of the sensors, \( S_n \), \( n = 1, \ldots, N \), is to generate status updates containing an observation of the physical phenomenon and a time-stamp. These sensors are deployed at locations \( x_n \), each transmitting status updates periodically at an interval \( T_u \). Transmitted status updates are collected at a gateway, i.e., a sink. An example of the considered system would be multiple sensors measuring temperature and sending their status updates to a thermostat.

We denote the AoI of the \( n \)th sensor, i.e., the time elapsed since the latest status update was received from the \( n \)th sensor, as \( \Delta_n(t) \). We assume that time is slotted, meaning that with each passing time-step, the AoI of the status update is increased by one time-unit. We determine the value of the AoI of a status update from the \( n \)th sensor at the sink at time slot \( t \) as in [3], by subtracting the generation time of the latest update \( U_n(t) \) from the current time as follows:

\[
\Delta_n(t) := t - U_n(t). \tag{1}
\]

In our system, sensors transmit status updates periodically. Consequently, the AoI for the \( n \)th sensor is constrained to an interval between 0 and \( T_u \).

3.1. Estimation in a system of correlated sensors

The main role of the sensing system is to provide accurate information regarding the value of the observed phenomenon at the location of interest to the service making decisions. We assume that all sensor locations are locations of interest. The system provides the required information by estimating the value of the phenomenon of interest using all gathered information [20]. For example, if a service requires information regarding the status of an observed phenomenon at the location of information sensor \( S_j \), i.e., \( Z(x_j, t) \), the system will use the \( \Delta_j \) aged information generated by \( S_j \) and fresher but spatially separated information collected by other sensors, i.e., \( S_i \) where \( i \in \{1, 2, \ldots, N \} \land \Delta_i < \Delta_j \), to estimate the requested value. We denote the spatial separation between the \( n \)th sensor and the \( i \)th sensor as \( d_{ni} \). To estimate the observed physical phenomenon, we employ a Linear Minimum Mean Square Error (LMMSE) estimator, which, as demonstrated in [32], offers a mathematically tractable way of estimating random processes in sensor networks.

We employ a covariance function to describe how the information in status updates from correlated sensors jointly varies over time and space. The covariance is a function of the spatial separation between two selected sensors and the time separations between status updates that they generate. The covariance enables us to determine how likely it is that a change in the value of the observed phenomenon at one location also transpired at another location. For example, the covariance reveals how likely it is that a change observed at sensor \( S_j \) also happened at \( S_i \) and vice versa. In our work, we adopt the following separable covariance model [33]:

\[
C|d_{ni}, t| \theta_1(t), \theta_2(t) = \exp(-\theta_2(t) d_{ni}) \theta_1(t) \Delta_n(t) - \Delta_i(t)). \tag{2}
\]

with \( \theta_1(t) \) and \( \theta_2(t) \) representing the scaling parameters of time and space respectively. To obtain the values of scaling parameters, we rely on Pearson’s correlation coefficient formula for samples, as established in [34]. Note that these scaling parameters change over time, and we periodically recalculate them using obtained observations. By establishing the correlation between the information sensors, we can determine the estimation error that results from using aged information from spatially separated sensors.

Every estimation of the observed physical phenomenon at the location of the \( n \)th sensor the system makes incurs an estimation error, which following the analysis in [36], we can express as:

\[
\varepsilon_{n}(x_n, t| \theta_1(t), \theta_2(t)) = 1 - c_{XY}(t) W_n(t). \tag{3}
\]

with covariance matrix \( c_{XY}(t) \) and \( W_n(t) \) being the LMMSE estimator weight vector, i.e., \( W_n(t) = [w_{n1}(t), \ldots, w_{nN}(t)]^T \). We can construct the covariance matrix by using the covariance model defined in Eq. (2). Using the estimation error we can determine how informative collected status observations are. By setting a threshold for the estimation error, we can control how timely collected information should be. In other words, the estimation error represents the quality of service (QoS) constraint the system has to ensure.

3.2. Lifetime of energy-constrained sensors

In the system considered here, sensors are powered either by a non-rechargeable battery or an EH solution. Sensors relying on a battery have limited expected lifetime, which depends heavily on the selected update interval. We model the expected lifetime of such sensors \( \mathcal{L}_n(T_u) \) as in [37]:

\[
\mathcal{L}_n(T_u) = \frac{E_B}{P_{step} + \frac{E_{sen}}{T_u}}, \tag{4}
\]

where \( E_B \) represents the battery-powered sensor’s initial energy, \( P_{step} \) is the sensor’s power consumption in a single time step, i.e., the energy sensors consume to stay operational, and \( E_{sen} \) is the expected energy the sensor requires to generate and transmit a new status update. On the other hand, the lifetime of EH sensors is not limited, due to their ability to replenish their energy. Unfortunately, such sensors may still experience downtime, i.e., time during which they do not have enough energy to transmit a fresh status update, i.e., when \( E_{sen}(t) < E_{sen} \). We assume that at each time step, the energy sensor will receive \( \rho_{sen}(t) \) amount of energy. The value of \( \rho_{sen}(t) \) depends on the current environmental value of the physical phenomenon that powers the energy harvester. In our numerical analysis we assume that sensors use a solar panel to collect energy; however, other phenomena may be
used by the harvester, such as wind or vibrations. Additionally, we assume that sensors have a limited capacity to store harvested energy, which we denote as \( E_H \). Therefore, an EH sensor \( n \)'s available energy is limited to an interval \( E_n(t) \in [0, E_H] \).

Next, we formulate the problem the system has to solve to ensure the timeliness of the collected information.

### 4. Problem description

The main objective of the system is to prolong the lifetime of battery-powered sensors while keeping the estimation error of status updates from every sensor below the set threshold. To achieve this objective, the system has to employ an updating strategy that will distribute the transmission of status updates among the sensors in proportion to their available energy, meaning that sensors with more energy will transmit more often. However, the energy of EH sensors varies significantly over time in a stochastic manner. For example, when a sensor relies on solar EH, the collected energy during the day will be much higher than during the night, or cloud cover might drastically lower the amount of harvested energy. Consequently, the system has to adjust its updating strategy to the time-varying nature of energy arrival on sensors powered by EH.

The system has to consider several factors when arriving at the optimum updating strategy. We show a high-level overview of the decision-making process in Fig. 2. When the gateway sets the next update interval for a sensor, it seeks to maximise the lifetime of the network, constrained by a bound on the estimation error associated with the desired information. The gateway has limited time available to make a decision due to the wake-up scheme energy-constrained sensors typically rely on. Typically, the gateway has around one second to respond.

The computational complexity of the problem rises as a polynomial of the number of sensors \( N \). The system has to calculate the estimation error for \( N \) sensors in future time-steps to ensure that the timeliness of information is maintained until the sensor’s next transmission. If the sensor transmits at time \( t_n \), the system has to keep the estimation below the set threshold at time-steps \( t_n + 1, t_n + 2, \ldots, t_n + T_n \). We limit the update interval to \( T_{\text{max}} \), which represents the longest update interval the system allows. In practice, the maximal update interval can be up to a few hours. The computational complexity of the problem at hand is \( O(N^4 T_{\text{max}}) \). Note that in many practical problems such a calculation can take a few seconds.

Due to the time-varying nature of the energy arrival on a sensor relying on EH and the high computational complexity of the problem, we base our proposed solution on DRL. DRL was proven to be able to adapt to time-changing nature of energy arrivals in numerous past works \([17,18,26–30]\). At the same time, we can leverage Artificial Neural Network (ANN) to determine the best possible action, i.e., the sensor’s update interval, in a matter of milliseconds, as only inference calculation based on the selected states is required. The proposed scheduler design is presented in detail in the next section.

### 5. Deep Reinforcement Learning based scheduler

By employing DRL, the system can deploy a learning agent, i.e., a scheduler, capable of determining the best updating strategy to prolong battery-powered sensors’ lifetimes while maintaining the accuracy of the collected information within a set target. The scheduler, which we implement using a DDPG algorithm \([39]\), resides in the gateway as we illustrate in Fig. 3, where it has access to all relevant information and computational power necessary to operate. We identified the DDPG algorithm to be the most suitable as it is aimed at DRL problems with continuous action spaces, as in the problem addressed in this paper; this is discussed in more detail later in this Section.

DDPG is an actor–critic DRL algorithm \([39]\). An actor–critic algorithm is a combination of value-based methods such as Q-learning, and policy-based methods, e.g., policy gradient. The actor part of the algorithm is in charge of taking actions based on the provided input state to find the optimal policy. The actor is implemented with a policy-based method. The critic’s role is to evaluate actions taken by the actor. The critic does so by calculating the value function for a particular action and state space. In addition, DDPG employs advances of deep learning introduced in recent years such as replay memory and use of target ANN during training. The DDPG algorithm is well suited for finding the optimal policy-updating strategy for the proposed problem, due to its ability to effectively approximate a large state space and work with a large action space, both present in our system.

#### 5.1. Scheduler state space, action set, and reward model

To provide the learning agent with complete knowledge of the environment, we describe the sensor’s state as a ten-dimension vector \( s_n \). The scheduler constructs state \( s_n \) from the perspective of the \( n \)th sensor that has just transmitted a status update as follows:

\[
\begin{align*}
\begin{pmatrix}
    E_{n1} \cdots E_{nN} \tilde{I}_1 \cdots \tilde{I}_N \eta \phi_1 \cdots \phi_j \\
    \bar{E}_{n1} \cdots \bar{E}_{nN} \bar{\eta} \bar{\phi}_1 \cdots \bar{\phi}_j
\end{pmatrix} 
= 
\begin{pmatrix}
    \prod_{i=1}^{N} E_{ni}^{\frac{1}{N}} \cdot \prod_{j=1}^{N} \bar{E}_{ni}^{\frac{1}{N}} \cdot \prod_{l=1}^{\eta} \tilde{I}_l \cdot \prod_{l=1}^{\bar{\eta}} \bar{\tilde{I}}_l \cdot \prod_{j=1}^{j} \phi_j \cdot \prod_{j=1}^{\bar{\phi}} \bar{\phi}_j \\
    \prod_{i=1}^{N} \bar{E}_{ni}^{\frac{1}{N}} \cdot \prod_{j=1}^{N} E_{ni}^{\frac{1}{N}} \cdot \prod_{l=1}^{\eta} \tilde{I}_l \cdot \prod_{l=1}^{\bar{\eta}} \bar{\tilde{I}}_l \cdot \prod_{j=1}^{\bar{\phi}} \bar{\phi}_j \cdot \prod_{j=1}^{\phi_j} \bar{\phi}_j
\end{pmatrix}
\end{align*}
\]

where \( \tilde{I}_l \) represent an identity function that takes a value of 0 when the sensor is powered by a battery or 1 if the sensor uses EH. The role of the identifier is to help the scheduler to differentiate between battery-powered and EH-powered sensors. Consequently, the scheduler can devise a different updating strategy depending on how the sensor is powered. Other information we identified as very important includes the ratio between the current estimation error and the set target, i.e., \( \tilde{I}_l^{1/\eta} \), the sensor’s update interval \( T_n \), energy level \( E_n \), and the energy collected in the last 24 hours \( \bar{E}_n \). Note that battery-powered sensors do not collect any energy, and thus \( \bar{E}_n = 0 \) for those sensors.

The first five dimensions of the state vector consist of the information related to the \( n \)th sensor, while the other five reveal information regarding the system in general, i.e., settings of other sensors in the system. To provide the scheduler with a good representation of the conditions on the majority of the sensors in the system we use the geometric mean.\(^4\) The use of the geometric mean (as opposed to individual values of the

\(^2\) The decision to consider sensors that use a solar panel is due to the availability of light intensity measurements from \([35]\). Using light observations, we can determine the amount of collected energy by following an analysis of a real solar-powered system examined in \([38]\).

\(^3\) The energy capacity for EH in real deployments is typically only a fraction of the capacity of a non-rechargeable battery, i.e., \( E_B \gg E_H \).

\(^4\) The geometric mean is defined as: \( \left( \prod_{i=1}^{N} u_i \right)^{\frac{1}{N}} = \sqrt[N]{u_1 \cdots u_N} \).
The scheduler can take an action to increase or decrease the sensor’s current update interval. The scheduler determines the next update interval in the following way:

\[ T_n = \min (\max(T_n' + [U_{\max}(\epsilon)], 1), T_{\max}). \]  

(6)

The \( a_\epsilon \) is the action value the scheduler receives from the actor’s ANN and is limited to an interval between \(-1\) and \(1\), i.e., \( a_\epsilon \in [-1, 1] \). In other words, the actor’s action is normalised. We multiply the received action value by \( U_{\max} \), representing the maximum change (increase or decrease) of the update interval we allow in the system. We round off the resulting product to obtain an integer number of time-steps. In the last step, we ensure that the newly determined update interval is realistically possible, i.e., no less than one, and that it does not exceed the maximum possible update interval \( T_{\max} \). As we demonstrate in the next section using real data, such an action setting is sufficient to solve the problem we posed in Section 4.

We tailored the reward to aid the learning agent in finding the sensors’ update intervals that lead to accurate sensing while simultaneously prolonging the lifetime of battery-powered sensors. To achieve the said goal, we identified three factors: the value of the estimation error, the sensor’s remaining energy, and the amount of energy collected in the past 24 h. By considering all three factors in our reward function, we can ensure that the scheduler can speedily find the optimal action value by \( U_{\max} \), representing the maximum change (increase or decrease) of the update interval we allow in the system. We round off the resulting product to obtain an integer number of time-steps. In the last step, we ensure that the newly determined update interval is realistically possible, i.e., no less than one, and that it does not exceed the maximum possible update interval \( T_{\max} \). As we demonstrate in the next section using real data, such an action setting is sufficient to solve the problem we posed in Section 4.

The reward associated with a sensor’s energy level is:

\[ r_{\text{col}}(E_n) = \begin{cases} 
1 - \frac{N E_n}{\sum_{i=1}^{N} E_i}, & \text{if } T_n > T_n' \\
0, & \text{if } T_n = T_n' \\
\sum_{i=1}^{N} E_i - 1, & \text{if } T_n < T_n'. 
\end{cases} \]  

(10)

The energy level reward encourages the scheduler to decrease the sensor’s update interval when the sensor has more energy available than others. Similarly, we constructed the reward term corresponding to the EH capabilities of the sensor as:

\[ r_{\text{col}}(\overline{\rho}_n) = \begin{cases} 
1 - \frac{N \overline{\rho}_n}{\sum_{i=1}^{N} \overline{\rho}_i}, & \text{if } T_n > T_n' \\
0, & \text{if } T_n = T_n' \\
\sum_{i=1}^{N} \overline{\rho}_i - 1, & \text{if } T_n < T_n'. 
\end{cases} \]  

(11)

The more energy the sensor has collected, the more likely the scheduler will be to decrease its update interval.

The Statistical Analysis module’s main role is to analyse correlation and store information relevant to the scheduling process. The gateway analyses and stores information regarding the sensors’ battery level, amount of energy collected, and the estimation error. Sensors periodically report their available energy levels to the gateway for EH sensors, the gateway can also calculate the amount of energy the sensor has collected from the measurements reported. The system uses the collected observations to extract the scaling parameters of time, i.e., \( \delta(t) \), and space, i.e., \( \beta(t) \), in the covariance function (Eq. (2)). The updated covariance function is then used to calculate the estimation error. To achieve a fast response time, the statistical analysis has to be carried out in parallel. The stored information enables the gateway to quickly determine the state and reward. Both are then passed on to the learning agent.

The Learning Agent is the heart of our scheduler. The DDPG algorithm consists of three major components: actor, critic, and experience replay
Algorithm 1 Energy-aware DDPG-based Scheduler

1. Randomly initialise critic network \(Q(s, a|\theta^Q)\) and actor \(\mu(s|\theta^\mu)\) with weights \(\theta^Q\) and \(\theta^\mu\).
2. Initialise target network \(Q'\) and \(\mu'\) with weights \(\theta'^Q \leftarrow \theta^Q\) and \(\theta'^\mu \leftarrow \theta^\mu\), replay buffer \(B\), and a random process \(N\).
3. Store experience \(s'_n, r_n, s_{n+1}, a_n\) in buffer \(B\).
4. Set initial action as \(a'_n = 0\) ∀ \(n \in \{1, 2, \ldots, N\}\).
5. Set initial update interval to \(T'_n = T_{start} \forall n \in \{1, 2, \ldots, N\}\).
6. while \(E_i(t) > 0\) ∀ \(n \in \{1, 2, \ldots, N\}\) do
7. if gateway receives status update from \(n\)-th sensor then
8. Determine the \(n\)-th sensors new state \(s'_n\) using Eq. (5).
9. Calculate reward \(r_n\) according to Eq. (7).
10. Store experience \((s'_n, a'_n, r_n, s_n)\) to \(B\).
11. Sample a random batch of \(M\) experiences from \(B\).
12. Set \(h_n = r_n + \gamma Q(s'_n, \mu'(s'_n|\theta'^\mu))\).
13. Update critic by minimising the loss function \(L\):
   \[
   L = \frac{1}{M} \sum_{i=1}^{M} \left( h_n - Q(s'_n, a'_n|\theta^Q) \right)^2
   \]
14. Update the actor’s policy:
   \[
   \nabla \theta^\mu \approx \frac{1}{M} \sum_{i=1}^{M} \nabla \mu(s|\theta^\mu) \mid_{s=s_i} \sum_{i=1}^{M} \nabla \mu(s|\theta^\mu) \mid_{s=s_i}\]
15. Update the target networks:
   \[
   \begin{align*}
   \theta'^Q &\leftarrow \theta^Q + (1 - \tau_C)\theta'^Q \\
   \theta'^\mu &\leftarrow \theta^\mu + (1 - \tau_A)\theta'^\mu
   \end{align*}
   \]
16. Select an action \(a_m = \mu(s_m|\theta^\mu) + \mathcal{N}\) and store it
17. Determine \(T_n\) \(n\)-th sensor using Eq. (6).
18. Transmit new update interval \(T'_n\) to \(n\)-th sensor
19. Update \(n\)-th sensor’s information gateway needs:
   \[
   a'_n \leftarrow a_n \\
   T'_n \leftarrow T_n
   \]
20. end if
21. end while

buffer. The scheduler passes the received state information to the actor. The actor determines the action using the learned policy. Note that the learning agent adds noise to the determined action. Adding the noise is important for a DDPG algorithm, as by randomly altering the determined action the algorithm explores the action space. For that, we use the Ornstein–Uhlenbeck process [40], a standard random process used for DDPG algorithm exploration we denote with \(N\). The role of the critic is to provide the actor with a grade, i.e., the output of the value function, which the actor uses to corrects its policy. The value function output is provided during the training of the actor’s and the critic’s ANN. During the training, the learning agent uses experiences stored in the experience replay buffer. An experience consists of the sensor’s current state, reward, the action taken in the previous state, and the sensor’s previous state.

In Fig. 3, the full line depicts the flow of activities, i.e., the steps the gateway takes and are detailed in Algorithm 1. At the start, the gateway initialises actor’s and critic’s policies, i.e., \(\mu(s|\theta^\mu)\) and \(Q(s, a|\theta^Q)\) respectively, along with target policies. Furthermore, the gateway also determines the initial state for each sensor, sets sensors’ update interval to \(T_{start}\), and start action values to zero. After the initialisation, the gateway will start listening for an incoming status update. Upon receiving a status update, the gateway will first determine the transmitting sensor state \(s_s\) and reward \(r_s\). Then, the gateway will store experience \((s'_s, a'_s, r'_s, s_s)\) in buffer \(B\). Then it will calculate the loss \(L\) of the random sample batch. With the obtained loss, the gateway can update the actor’s policy using the sampled policy gradient in step 14. In the next step, the gateway performs a soft update and takes action \(a_n\). Using the obtained action value, the gateway determines the new update interval with Eq. (6). In the last step, the gateway will store information it needs to determine the transmitting sensor state and reward the next time the gateway will receive an update from the \(n\)-th sensor. Note that while the gateway is making the decision, i.e., performing steps 8 to 20, the sensor is in hibernation mode. For example, a sensor using a LoRaWAN radio operating in class A will enter hibernation mode for one second. In our implementation, the response (a command with the sensor’s new update interval) is ready within 10 ms from the time the gateway receives a status update.

6. Validation and results

In this section, we evaluate the performance of the proposed scheduler using real data published by the Intel Berkeley Research laboratory [35]. In the first subsection, we explain how we use data to model the environment with which our scheduler interacts. We also describe how sensors collect energy using EH, and list other energy-related parameters. In the second subsection, we evaluate the performance in terms of the achieved lifetime and compare it to a baseline. We also explore the impact of \(e^*\), average distance between sensors, and the amount of daily collected energy, i.e., \(p_e\), on the overall performance.

We implemented the DDPG algorithm using PyTorch [41], a python based library for Machine Learning (ML). The ANNs of both the actor and the critic consist of four feed-forward hidden layers. The first three hidden layers consist of 75 neurons, while the fourth one has 25 neurons. We apply a ten percent dropout between hidden layers to prevent over-fitting. We use the ReLu activation function for every layer except for one. The actor’s output layer is activated with a Hyperbolic function.

As most contemporary DRL solutions, the actor and critic use target and evaluation ANNs. The target ANNs are used during the training process to prevent over-fitting of the learned behaviour to the current batch of experiences. The batch, in our implementation, consists of 128 randomly selected experiences from a buffer of size 100,000. To obtain the DRL parameters, we performed a grid search and arrived at values we list in Table 1 as most suitable.

6.1. Modelling the environment using real data

To assess the performance of our scheduler, we employ real measurements obtained by 54 sensors deployed on one floor in the Intel Berkeley Research laboratory [35]. These real measurements of temperature, humidity, and light intensity represent the ground truth in our simulated environment. We iterate the simulation over nine days in time-steps of 10 s. This means that our simulation goes though 77,760 unique time-steps. For each time-step, we have a measurement the real sensor has collected. Note that in the real deployment sensors transmitted every 31 s: to obtain the values in the missing time-steps we used linear interpolation. Each sensor is assigned a unique identifier. The gateway receives a status update, i.e., measurement, only from sensors that successfully transmitted in the given time-step.

In Table 2 we list the energy parameters we selected by following trends in real IoT sensor deployments. We assume that battery-powered sensors are powered by a standard button cell battery. Normally such batteries have a capacity of 200 mAh. However, to shorten the simulation time, we assume that battery capacity is only 5 mAh. In our simulation, we assume that EH sensors use a 10 F capacitor. We determined the energy that the sensor requires to transmit, \(E_{tr}\), based on the analysis of LoRaWAN in [42]. The value for the continuous power consumption of sensors, \(P_{stop}\), is based on the power consumption of sensors used in Intel deployment.

We model the EH sensors’ ability to collect energy using the light intensity measurements provided by the Intel dataset, coupled with solar panel analysis provided by Yue et al. [38]. In [38], the authors...
6.2.1. The effect of EH-powered sensors on battery-powered sensors’ life-
resulting AoI. In the last part, we examine the impact of the average ones. Then we proceed to ascertain the impact of target estimation error to determine the influence of EH-powered sensors on battery-powered of EH-powered sensors in the system. Such an approach enables us system. First, we measure the expected lifetime as we vary the number parts. In each part, we explore different characteristics of the proposed a sensor makes per day, and AoI. We split the assessment into three expected lifetime of battery-powered sensors, number of transmissions the training process.

\[ \text{expected lifetime} = \frac{\text{energy stored}}{\text{energy consumption per update}} \]

measured the output current of a solar panel and mapped it to the Intel dataset with the model in [38] we can determine how much energy an EH sensor can collect in every time-step. However, the measurements from [38] are only available up to 1000 luminance (lux) units. Consequently, we limit the amount of energy a sensor can collect in a given time-step. For illustration, if a sensor was exposed to the maximal value of light intensity for the entire day the sensor would collect 14.5 J.

6.2. System performance

To assess the performance of our DRL-based scheduler, we split the available data into two parts. The first part consists of three days of data, and we use it to train the scheduler. We use the other six days for the validation. Such a split ensures that the learned behaviour is not over-fitted to the environment. Furthermore, we added a fail-safe mechanism: Suppose that the value of estimation error exceeds \( \varepsilon^* \) for more than 25 percent of the time. In that case, the agent takes an automatic action to reduce the sensor’s maximum update interval increment \( U_{\text{max}} \). Such occurrences are rare and occurred only during the training process.

We assess the system by measuring three key performance metrics: expected lifetime of battery-powered sensors, number of transmissions a sensor makes per day, and AoI. We split the assessment into three parts. In each part, we explore different characteristics of the proposed system. First, we measure the expected lifetime as we vary the number of EH-powered sensors in the system. Such an approach enables us to determine the influence of EH-powered sensors on battery-powered ones. Then we proceed to ascertain the impact of target estimation error (\( \varepsilon^* \)) on the number of transmissions a sensor makes each day and the resulting AoI. In the last part, we examine the impact of the average distance between sensors on the number of daily transmissions.

6.2.1. The effect of EH-powered sensors on battery-powered sensors’ life-
time

In the first experiment, we vary the number of EH sensors from 5 to 45, while the total number of sensors in the system remains the same (50), and measure the expected lifetime of the system. At the start of each simulation, we randomly select which sensors are battery-powered and which use EH. Consequently, we have to run multiple instances of each simulation setting to remove the effects of sensors’ location on the results. For example, if EH sensors are located near the window, the system achieves longer lifetime than when all EH sensors are located in the shade. We run the experiment 10 times for each simulation setting and report the average lifetime obtained. We follow this approach for both our solution and the genie-aided one. Note that every simulation lasts until the first battery-powered sensor consumes all its available energy. To provide the reader with a realistic feel for how long such sensor deployments can last, we scale the obtained results, as real battery-powered sensors would be powered by a battery with a 200 mAh capacity.

We compare the performance of our scheduler to two other scheduling schemes:

1. **Best Uniform Approach (BUA):** Represents the lifetime sensors can achieve by adopting the best uniform update interval. This is the maximum constant update interval that, if adopted by all sensors, will ensure the estimation error constraint \( \varepsilon^* \) is never exceeded. Note that this approach is not energy-aware.

2. **Energy-aware Genie:** In this approach, the battery-powered sensors transmit only when the estimation error constraint is about to be violated. On the other hand, EH sensors also transmit when their energy level is greater than the average energy level of battery-powered sensors. Additionally, we assume that our genie can wake any sensor up on demand without any additional energy cost to the sensor. This means that the genie has an advantage over our approach because our scheduler has to decide on the sensor’s update interval up front, i.e., within the period when the sensor listens to the transmission channel.

Fig. 4 shows the lifetime that the system can achieve when observing temperature, for our DDPG-based approach and for the two baseline schemes described above. Interestingly, the performance of the DDPG-based scheduler is on par with the genie-aided one. Additionally, battery-powered sensors’ lifetime increases with the number of EH-powered in the system. For example, the lifetime increases by 15% when the number of EH-powered sensors increases from 5 to 45. Furthermore, the lifetime more than doubles in comparison to the best uniform approach. Similarly, when the system observes humidity the lifetime also more than doubles when using our proposed scheduler in comparison to the best uniform approach. We show the humidity-related results in Fig. 5. Note that for that case the genie-aided baseline performs significantly better. The difference in the two sets of results comes from the different behaviour of the observed phenomenon. We speculate that both are influenced by the actions that have happened in the laboratory when measurements were collected and analyse the dataset in greater depth to illustrate this.

In Fig. 6 we show the change in temperature and humidity sensors over three days. Each morning, the temperature sharply rises while the relative humidity decreases rapidly. Around midday, the temperature starts falling, and it decreases until the following morning. On the other hand, the changes in relative humidity measures are minor and remain relatively constant until the next morning. We speculate that the lab personnel opens windows in the morning. An opened window could explain the rapid change in the humidity, while the temperature is not as affected. Typically humidity is in the range of 40–50 percent. Then, in the morning, it decreases by around 20–30 percent in an hour. The DDPG-based scheduler is very good at following gradual changes, as it achieves almost optimal performance, compared against the genie-aided solution, for the temperature measurements. For more significant swings in the measured values followed by a constant value, as is the case for the humidity readings on day one, it is to be expected that a genie, with complete information about the future, can outperform our solution, which relies on learning.

Next, we analyse the proposed scheduler’s computational complexity and sampling efficiency.
The change of expected lifetime for a system observing temperature, as the number of sensors powered by an energy-harvester increases. We set $\varepsilon^*$ to 2%.

The change of expected lifetime for a system observing humidity, as the number of sensors powered by an energy-harvester increases. We set $\varepsilon^*$ to 2%.

The change of temperature and relative humidity as measured by sensors over three days in Intel Berkeley Research laboratory. The starting time is on midnight, March 1st 2004.

6.2.2. Computational complexity and sampling efficiency

The centralised approach, i.e., collecting and processing information on the gateway, has two benefits that help the proposed scheduler overcome computational complexity challenges. The first benefit is available computational power that enables the scheduler to perform all necessary calculations to obtain the decision within 10 ms. The second benefit is combining experiences from all sensors in the same replay buffer $B$. In such an approach, the scheduler leverages information obtained by one sensor to speed the learning process for every sensor in the system. Furthermore, in our approach, each received status update represents one experience. Therefore, the scheduler requires time to collect necessary experiences. For example, as we show in more detail in a later Section 6.2.4 on a specific use case with $\varepsilon$ set to two percent, on average, a sensor will transmit a status update 100 times a day. Such an updating rate translates into 5000 samples, i.e., experiences, the gateway will receive each day. However, as we show next, the collected number of samples is sufficient to enable the agent to reach the desired behaviour relatively quickly after the initialisation.

In Fig. 7 we show the number of samples the scheduler requires to collect in our evaluation case study before it is capable of learning to set the sensor’s update interval to the optimal value for different $\varepsilon^*$. We consider the scheduler performs optimally when it is capable of adapting sensors’ update intervals in such a way that the estimation error is close, but never more than the set accuracy constraint, i.e., $\varepsilon^*$. Additionally, in our case study, we observed that the scheduler, after this initial training process, it can adapt to dynamic changes in the system. For lower values of $\varepsilon^*$, the scheduler requires more samples for the initial training than for higher values of $\varepsilon^*$. To a certain degree, such behaviour is expected as with lower $\varepsilon^*$ the sensor needs to transmit more frequently, which leads to lower sample efficiency. Interestingly, the scheduler will collect required samples faster when $\varepsilon^*$ is lower.

6.2.3. Target estimation error

Next, we analyse the performance of our scheduler as we vary $\varepsilon^*$. Fig. 8(a) shows that the increase in the expected lifetime is essentially linear in $\varepsilon^*$. Fig. 8(b) shows the median number of times that battery-constrained and EH sensors transmit per day. Interestingly, both groups perform a similar number of transmissions. The only noticeable difference is at a very low $\varepsilon^*$. At very high accuracy targets, the limited energy harvesting capacity of EH sensors limits the number of transmissions those sensors can make, and therefore the battery-powered sensors have to transmit more often to compensate. Finally, in Fig. 8(c) we plot the number of transmissions per day for the sensor with the most transmissions. We can see that the worst-case number of transmissions for the battery-powered sensors is reduced much faster than for an EH sensor. Such behaviour indicates that the system tries to use the energy from EH sensors that collect more energy to preserve the lifetime of battery-powered ones.

This is the average time the proposed scheduler required to provide a response in our test environment with Intel® Core™ i5-8279U CPU and Python version 3.9.
The change in network lifetime, the median number of transmissions sensors make per day, and the maximum number of transmissions a sensor makes per day, as a function of the target estimation error ($\epsilon^*$). The results are obtained in a system with 25 sensors powered by a non-rechargeable battery and 25 EH sensors.

Fig. 9. The change of average AoI as a function of the target estimation error ($\epsilon^*$). The results are obtained in a system with 25 sensors powered by a non-rechargeable battery and 25 EH sensors.

The average AoI of collected status updates increases with $\epsilon^*$ as we show in Fig. 9(a) and Fig. 9(b). The trend is noticeable for both temperature and humidity. Furthermore, the average AoI depends on how sensors are powered. When the $\epsilon^*$ is high, the average AoI of status updates from battery-powered sensors is higher in comparison to the average AoI of status updates from an EH powered sensor. Such behaviour can be attributed to the excess energy the EH sensors have, which can then, in turn, be used to compensate for the outdated information from battery-powered sensors while still ensuring that the $\epsilon^*$ target is met. The peak AoI, which represents the worst-case possible in terms of AoI, is on average higher for EH powered sensors, as shown in Fig. 10(a) and (b). The peak AoI is higher for EH powered sensors as some collect very little energy, e.g., a sensor placed in the shade. Thus status updates from those sensors result in a higher peak AoI.

Fig. 10. The change of peak AoI as a function of the target estimation error ($\epsilon^*$). The results are obtained in a system with 25 sensors powered by a non-rechargeable battery and 25 EH sensors.

6.2.4. Distance between sensors

In general, the closer to other sensors the sensor is, the fewer transmissions it will make. Figs. 11 and 12 show such a trend for both battery-powered and energy-harvesting sensors. However, we can see that the increase in daily transmissions also depends on how sensors are powered. The slope is more gradual for battery-powered sensors than for energy-harvesting ones. Such behaviour might come as a surprise as the agent’s state space, presented in Eq. (5), does not contain information capturing the spatial correlation or distances between the sensors. Nonetheless, the agent appears to capture and exploit the fact that the sensors are correlated, even though it cannot attribute it directly to location proximity.

The scheduler tends to select one sensor, usually one capable of collecting more energy, to transmit more often than others. In such a way, the sensor transmitting more often enables other EH sensors to recuperate their energy. However, the spatial correlation, i.e., the distance to other sensors, impacts how many transmissions will sensor make in a day.
7. Conclusion

In this paper, we used statistical analysis to determine correlation and employed a DRL at the gateway to design a scheduler capable of prolonging battery-powered sensors’ lifetime. By taking advantage of more timely information from correlated sensors powered by an EH, we can improve the sustainability of IoT deployments. Our validation, carried out using real data from a sensor network deployment, showed that our proposed approach could provide more than double the life expectancy of battery-powered sensors, compared to a non-adaptive method. Furthermore, its performance can, in some instances, rival that of an idealised genie-aided scheduler. The combination of data analytics and DRL cannot be overstated. As we demonstrated in our work, these techniques are powerful tools in building self-sufficient and self-sustaining networks of the future.

As expected, two sensors have more correlated measurements when placed in a similar site, e.g., sensors placed close to a window. In other words, the sensors with the same macro-environment, e.g., window location, tend to be more correlated. Gaussian Processes (GP) [43] could be employed to take advantage of such correlation. Using GP, we would be able to model the correlation using kernel functions (covariance functions) even more effectively as we could also include metainformation. Additionally, we can easily alter the proposed solution to accommodate the use of GP. The main change would be to replace the measure of estimation error with the value of variance resulting from the use GP estimation in the state space and the reward function. However, the downside is that GP requires substantial computational resources to operate.

CRediT authorship contribution statement

Jernej Hribar: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. Luiz A. DaSilva: Conceptualization, Methodology, Validation, Formal analysis, Writing review & editing, Visualization, Supervision, Funding acquisition. Sheng Zhou: Conceptualization, Methodology, Validation, Writing – review & editing. Zhiyuan Jiang: Conceptualization, Methodology, Validation, Writing – review & editing. Ivana Dusparic: Conceptualization, Methodology, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded in part by the European Regional Development Fund through the SFI Research Centres Programme under Grant No. 13/R/2077_P2 SFI CONNECT and by the SFI-NSFC Partnership Programme Grant Number 17/NSFC/5224. It is also supported by the Commonwealth Cyber Initiative at Virginia Tech.

References

[2] B. Yin, S. Zhang, Y. Cheng, L.X. Cai, Z. Jiang, S. Zhou, Z. Niu, Only two sensors have more correlated measurements when placed in a similar site, e.g., sensors placed close to a window. In other words, the sensors with the same macro-environment, e.g., window location, tend to be more correlated. Gaussian Processes (GP) [43] could be employed to take advantage of such correlation. Using GP, we would be able to model the correlation using kernel functions (covariance functions) even more effectively as we could also include metainformation. Additionally, we can easily alter the proposed solution to accommodate the use of GP. The main change would be to replace the measure of estimation error with the value of variance resulting from the use GP estimation in the state space and the reward function. However, the downside is that GP requires substantial computational resources to operate.

CRediT authorship contribution statement

Jernej Hribar: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. Luiz A. DaSilva: Conceptualization, Methodology, Validation, Formal analysis, Writing review & editing, Visualization, Supervision, Funding acquisition. Sheng Zhou: Conceptualization, Methodology, Validation, Writing – review & editing. Zhiyuan Jiang: Conceptualization, Methodology, Validation, Writing – review & editing. Ivana Dusparic: Conceptualization, Methodology, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was funded in part by the European Regional Development Fund through the SFI Research Centres Programme under Grant No. 13/R/2077_P2 SFI CONNECT and by the SFI-NSFC Partnership Programme Grant Number 17/NSFC/5224. It is also supported by the Commonwealth Cyber Initiative at Virginia Tech.

References


