Collaborative duty cycling strategies in energy harvesting sensor networks

Received: 00 Month 0000 | Revised: 00 Month 0000 | Accepted: 00 Month 0000

DOI: xxx/xxxx

ARTICLE TYPE

Collaborative duty cycling strategies in energy harvesting sensor networks

James Long | Oral Büyüköztürk*

[1]Department of Civil and Environmental Engineering, Massachusetts Institute of Technology, Massachusetts, USA

***Correspondence**
Oral Büyüköztürk, 77 Massachusetts Avenue, Cambridge, MA 02138 USA Email: obuyuk@mit.edu

Summary

Energy harvesting wireless sensor networks are a promising solution for low cost, long lasting civil monitoring applications. But management of energy consumption is a critical concern to ensure these systems provide maximal utility. Many common civil applications of these networks are fundamentally concerned with detecting and analysing infrequently occurring events. To conserve energy in these situations, a subset of nodes in the network can assume active duty, listening for events of interest, while the remaining nodes enter low power sleep mode to conserve battery. However, judicious planning of the sequence of active node assignments is needed to ensure that as many nodes as possible can be reached upon the detection of an event, and that the system maintains capability in times of low energy harvesting capabilities. In this paper, we propose a novel reinforcement learning agent which acts as a centralised power manager for this system. We develop a comprehensive simulation environment to emulate the behaviour of an energy harvesting sensor network, with consideration of spatially varying energy harvesting capabilities, and wireless connectivity. We then train the proposed reinforcement learning agent to learn optimal node selection strategies through interaction with the simulation environment. The behaviour and performance of these strategies are tested on real unseen solar energy data, to demonstrate the efficacy of the method. The deep reinforcement learning agent is shown to outperform baseline approaches on both seen and unseen data.

KEYWORDS:
structural health monitoring, reinforcement learning, energy harvesting, event detection, duty cycling

1 | INTRODUCTION

Low power, low cost energy harvesting wireless sensor networks promise to dramatically reduce instrumentation costs and drive widespread adoption in monitoring of structural health, seismicity and environmental conditions. These systems are comprised of many individual nodes, each with computing capability, wireless communication, battery power and one or more sensors. A typical battery powered sensor node can last years in the lowest power sleep mode, but only days when fully active. When in low power sleep mode, sensor nodes have limited capability, and designing a system which balances immediate functionality with longevity is a difficult task.

Many structural and vibration monitoring applications are fundamentally concerned with the detection and recording of specific events, rather than the continuous collection of data. Events of interest in structural health monitoring applications include seismic activity, periods of high wind, heavy vehicle loading or user-initiated requests for data. To conserve energy, a subset of nodes can be kept active, listening for events, while the remaining nodes sleep. On detection of an event, active nodes trigger sleeping nodes to wake up and record data. This

scheme is implemented by Rice et al. (2010), as part of the Illinois Structural Health Monitoring Project software toolset. However, active nodes deplete their battery quickly, and so are generally either continuously powered, or adopt a low duty cycle. For example Jang et al. (2010), deploy a bridge monitoring system where 'sentry' nodes record acceleration and wind data for 10 seconds every 10 minutes, triggering a system wide wakeup upon exceeding a predetermined threshold. Actively listening nodes are frequently referred to as sentry nodes in the literature; well will use the term active node and sentry node interchangeably in this paper.

Both Liu, Cao, Tang, & Wen (2016), and Jang et al. (2010), describe battery powered wireless sensor systems, in which battery reserves monotonically decrease over time. Energy harvesting capabilities can be a relatively cost effective addition to sensor nodes, extending node lifetime and achievable duty cycle levels. Many sources of energy, especially solar, exhibit strong temporal variations, which lead to fluctuations in battery levels. This presents unique challenges in planning and scheduling the behaviour of sensor nodes. The increased availability of energy enables battery powered sentry nodes with much higher duty cycles than in non-energy harvesting networks. But these sentry nodes deplete energy quickly in periods of low energy availability, while sleeping nodes maintain battery reserves. Rotating the set of active nodes based on available battery reserves is an obvious solution to this problem, but careful planning of the sequence of active nodes is needed to ensure network performance meets service requirements during times of low energy availability.

In this paper we treat sentry node selection as a sequential decision making problem, and propose a centralised reinforcement learning solution. In this approach, nodes report their state at regular intervals to a centralised power manager, which returns instructions to alter the set of sentry nodes if necessary. The power manager aims to maximise the average number of reachable nodes in the network over time: Nodes are reachable if they are within a single hop of a sentry node. Unlike Liu et al. (2016), we do not consider multi-hop wakeup procedures. Casting this problem as a sequential decision making problem ensures that the centralised power manager explicitly accounts for tradeoffs between current and future utility, balancing between actions which have immediate reward (assume sentry node duty), and delayed reward (preserve battery for future sentry node duty). This is especially useful in solar energy harvesting nodes, which exhibit strong temporal variations in available energy.

A comprehensive simulation environment which allows for realistic modelling of sensor node energy consumption, recharging capability and communication is developed to train and evaluate the proposed reinforcement learning approach. This simulation environment adheres to the OpenAI Gym (Brockman et al., 2016) framework, which provides a standardised interface to allow modular testing of different learning and control agents. The reinforcement learning agent is then trained through interaction with this simulation environment. Finally comparison with several baselines adapted from existing literature is conducted, and analysis and visualisation of the learned strategies is provided.

This approach is intended for structural health and vibration monitoring applications which share two important characteristics: 1) Events of interest occur relatively infrequently, and data recorded outside of these events is of little or no value, 2) One node, or a small subset of the total nodes in the network, can reliably detect the onset of an event of interest. As a result, we do not consider the problem of minimising energy consumption arising from data transmission, nor do we consider in this paper the effect of sentry node selection on the likelihood of event detection.

2 | RELATED WORK

A related problem arises in hierarchical sensor networks, where networks are comprised of clusters of nodes, each with a cluster head node, which collects and potentially aggregates data from other nodes, before forwarding it to a central server, e.g. Gao, Spencer Jr, & Ruiz-Sandoval (2006), Sim, Carbonell-Márquez, Spencer Jr, & Jo (2011). In these hierarchical networks, the number and location of cluster heads impacts the energy efficiency of the system as a whole. Extensive research has been conducted on developing algorithms which find data transmission routes which minimise energy consumption and/or balance energy consumption between nodes, e.g. Shah & Rabaey (2002), S.-C. Huang & Jan (2004). Unlike this paper, these algorithms are fundamentally concerned with managing energy consumption arising from data transmission, with the assumption that sensor nodes are continuously sensing data which must be collected at a base station. In SHM specific applications, there is often an additional tradeoff between the energy efficiency of the sensor network layout, and the quality of the modal information obtained from measurements. Fang, Liu, & Teng (2018) derive an analytical expression for the maximally energy efficient number of clusters in two layer hierarchies comprised of i) cluster head (router) nodes, and ii) leaf nodes. A genetic algorithm is then developed to choose the number of sensors and sensor spacing which achieves the best tradeoff between energy efficiency and modal information quality. Fu, Ghosh, Johnson, & Krishnamachari (2013) propose an energy efficient routing tree for grid layouts of sensor nodes with centrally located base stations and multiple layers, and observe that a tradeoff between energy efficiency and modal information quality arises as a function of the grid

size: For closely spaced configurations data can be transmitted at low cost, but modal information quality deteriorates as a smaller total area is covered. In this work we assume a predetermined fixed sensor network layout, and for simplicity do not consider the modal information quality obtained from the network.

Energy-efficient configurations, even when they are specifically designed with energy-balancing in mind, can still lead to situations where nodes in high-traffic locations deplete their battery more quickly than others. Adaptive clustering approaches, which periodically rotate node assignments between cluster head duties and non-cluster head duties, can ameliorate this issue. Heinzelman, Chandrakasan, & Balakrishnan (2002), propose a method for rotating cluster head duties called low energy adaptive clustering hierarchy (LEACH). In the centralised version of this approach, cluster head duties are chosen at discrete time intervals by a simulated annealing procedure which minimises the summed squared distances between non cluster head nodes and their closest cluster head node. Non cluster head nodes are then assigned a specific time slot to transmit data in a time-division multiple access scheme, outside of which they may conserve energy.

Xiao, Zhang, & Dong (2013) propose a modified version of LEACH for application in sensor networks with energy harvesting capabilities. In Xiao et al. (2013), an energy potential function is defined for each node, describing its energy harvesting capabilities. This function is then used in selection of cluster head nodes, increasing the likelihood of selecting those with high energy harvesting capabilities. Bahbahani & Alsusa (2017) develop another adaptation of LEACH, in which duty cycle levels between 0 and 1 are set for each node, describing what proportion of time they spend as a cluster-head, and what proportion of their available time slots they utilise for data transmission when not a cluster head. Each nodes cluster-head duty cycle is set so that the expected number of cluster-heads minimises the total energy consumption of the network. Data transmission duty cycles are adjusted to ensure that each node's energy consumption matches the amount of energy it harvests: a condition called energy neutral operation (ENO).

Achieving energy neutral operation of sensor nodes is itself the focus of a significant number of studies. Previous work has focused on adapting individual nodes duty cycle levels so that the amount of energy consumed by the node is less than the amount of energy harvested. The maximum duty cycle level which still maintains energy neutral operation is termed node level energy neutrality, or ENO-max, as defined by Kansal, Hsu, Srivastava, & Raghunathan (2006).

Various different algorithmic approaches have been proposed to solve the problem of achieving node level energy neutrality. Kansal et al. (2006) formulate a linear programming approach which, combined with a predictive model of energy harvesting, solves for the highest possible duty cycle that meets the criteria of energy neutral operation. Energy harvesting predictions are made by adaptively fitting an exponentially weighted moving average model to measurements of harvested energy. Vigorito, Ganesan, & Barto (2007), proposed a linear quadratic tracking algorithm, which unlike the approach in Kansal et al. (2006), does not require a model of future energy generation.

More recently, reinforcement learning (RL) algorithms have been proposed as a more flexible and powerful approach to achieving node level energy neutrality in energy harvesting sensor networks. In Chan et al. (2015) the challenge of maximising quality of service while maintaining battery reserves is framed as a continuous time markov decision process, and solved using tabular Q-learning. Hsu, Liu, & Wang (2014) propose a tabular Q-learning approach for query driven wireless sensor networks. This approach assumes that a certain throughput level is demanded by a query to a sensor network, and aims to meet this demanded throughput while maintaining energy neutral operation. Shresthamali, Kondo, & Nakamura (2017), propose a similar approach, but using a tabular SARSA algorithm for reinforcement learning, and incorporating weather forecast data to improve performance.

Research on energy neutral operation of sensor nodes has focused on the individual node level, and omitted considerations of the resulting impacts on network level performance. Achieving the ENO-max condition at each node individually may lead to suboptimal network level behaviour. If 0 % duty cycle periods occur simultaneously at all nodes in the network, any events occurring in this period will go undetected. Tabular reinforcement learning algorithms as proposed in (Chan et al., 2015), (Hsu et al., 2014), (Shresthamali et al., 2017) are likely to be intractable when considering the entire network, due to the size of the state-action space, and so more powerful reinforcement learning approaches are required, which utilise function approximators instead of lookup tables, as in (Mnih et al., 2015), (Mnih et al., 2016), (Peters & Schaal, 2008). Deep neural network based approaches have also been proposed for civil and structural monitoring problems recently, outside the context of reinforcement learning, as in Y. Huang, Beck, & Li (2018), Rafiei & Adeli (2017), and Rafiei & Adeli (2018).

3 | PROBLEM STATEMENT

The goal of this paper is to develop a strategy which maximises the utility of the overall system by intelligently selecting which subset of nodes is active at each period in time. This problem is fundamentally one of sequential decision making under uncertainty, and can naturally be cast as a reinforcement learning problem. In the following section we will first

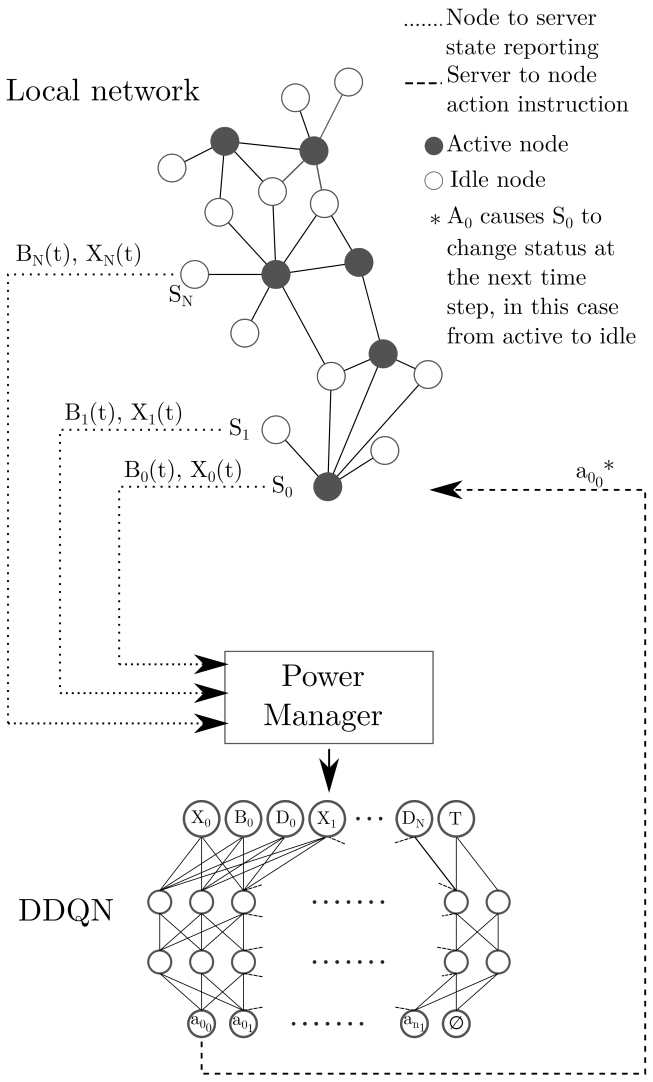

FIGURE 1 Overall schematic of the proposed system architecture, showing the peer to peer communication in the local network, and remote node to power manager communication used for reporting state and action instructions

describe the overall architecture of the wireless sensor system, before providing details used in simulating the power consumption, recharging capabilities and wireless connectivity of nodes in the network. In Section 5, we will introduce the proposed reinforcement learning framework, and formulate our problem in this framework.

We assume in this problem that events during which recorded data is of high value occur infrequently, are of relatively short duration, and that any information outside of these event windows is of little to no value. These assumptions are not valid in all structural monitoring applications, but are reasonable in most seismic monitoring applications, bridge and structural monitoring applications where high amplitude excitations occur due to rail, very heavy vehicle or high wind loading, or wireless sensor network applications where requests for data are initiated by users. We additionally assume that events of interest can be reliably detected by any node in the network. This limits the scope of this approach to networks of moderate size, and events which are non-local in nature, both commonly valid in structural and vibration monitoring applications.

4 | SYSTEM MODEL

Our model is comprised of n individual sensor nodes, which together make up the set G. We assume two distinct modes of communication in the network: 1) Node to remote power manager, which is used for infrequent, time insensitive messaging, in this case, communicating the current status and battery life and receiving action instructions. The node to manager connection can be achieved through cellular LTE or LoRa connection, as implemented for example in Addabbo et al. (2019). 2) Peer to peer communication, which can be achieved via Bluetooth or Zigbee radio and is used for time sensitive, low volume messaging, for example, triggering a neighboring node to wake up and record an event of interest. A schematic of this system is shown in Figure 1 . At regular intervals, each node in the network reports its state directly to the power manager. The power manager then publishes action to nodes as necessary, allowing the set of active nodes to change over time. Active nodes maintain local peer-to-peer connections which are used to trigger a system wide wake up when an event of interest is detected.

Each individual node in our network model is equipped with a battery, a processor, a radio, a physical sensor, and a means of harvesting energy from its environment. In this section we will provide details on the battery and power consumption characteristics of each node, recharging capabilities, and wireless connectivity properties.

4.1 | Node energy consumption

Every node has a fixed, and identical battery capacity, denoted B_{max}. We consider a discrete time model, in which each discrete time slice is denoted t_k, and has duration Δ_t. The battery reserve of node i at the onset of time slice t_k is given by $B_i(t_k)$ For every period t_k, the status of each sensor is given by $X_i(t_k)$ where $X_i \in \{0, 1, 2\}$, where $X_i = 0$ indicates that node i is in deep sleep mode, $X_i = 1$ indicates that it is in idle mode, and $X_i = 2$ indicates the node is active. Individual sensor nodes consume available power at a fixed rate P_a while active, a lower rate P_i when idle and a much lower rate P_s, while in deep sleep mode.

The active energy consumption rate of a sensor P_a is calculated by summing the active power consumption of the sensor node microcontroller, the radio, and the MEMS accelerometer. In idle mode, the processor board and MEMS accelerometer are deactivated. The peer to peer radio remains in active mode, listening for a signal from a neighboring node to wake up. P_i is therefore calculated by summing the idle mode power consumption rate of the microcontroller and the MEMS accelerometer, and the active mode power consumption rate of the radio. P_i is dominated by the active mode power consumption rate of the radio.

Although idle nodes can be triggered to wake up and record data when an event occurs, we neglect the energy consumed during this potential brief period of activity. As noted in Section 1, we assume that events of interest occur relatively infrequently, and are of relatively short duration, and therefore the total time by an idle node in recording data during any one discrete time period is insignificant. For applications in which events of interest occur frequently, or are of long duration, this assumption may not be valid.

Deep sleep mode consumes the least amount of power, but severely limits the functionality of the node. In deep sleep mode, the radio, processor and sensor all enter their lowest power consumption state. The processor can exit this state through a scheduled interrupt, allowing it to then wake the radio and sensor using pin interrupts. Nodes are uncommunicative for the entire time period when in deep sleep mode.

Specific values used in this study are given in Table 1, but the methodology presented can be adapted without any loss of generality. For reference, the chosen battery capacity B_{max} used in simulations in this paper is 2000 mAh, with an average discharge voltage of 3.7V.

	Deep sleep (mW)	Idle (mW)	Active (mW)
Processor	0.185	1.85	203.5
Radio	0.03	166.5	166.5
Sensor	0.185	1.85	55.5
Total	0.4	170.2	425.5

The total energy consumed in period t_k, $E_c(t_k)$ can be written as follows:

$$E_c(t_k) = \begin{cases} P_s \cdot \Delta_t & \text{when } X_i(t_k) = 0 \\ P_i \cdot \Delta_t & \text{when } X_i(t_k) = 1 \\ P_a \cdot \Delta_t & \text{when } X_i(t_k) = 2 \end{cases} \quad (1)$$

4.2 | Energy harvesting

The energy consumed by sensor i in time period t_k, $E_{c_i}(t_k)$, is offset by the energy harvested $E_{h_i}(t_k)$, such that the battery level at time $t+1$, can be calculated as follows:

$$B_i(t_{k+1}) = B_i(t_k) + E_{h_i}(t_k) - E_{c_i}(t_k) \quad (2)$$

The harvested energy is unlikely to be constant across each node in the network. Even if the energy harvesting capabilities of each node are identical, for most energy harvesting systems we expect spatial variation in the available energy. To model this effect, we represent the available energy over the area of the sensor network as a random field. The energy harvested over the entire area of the sensor network has a baseline mean value for each time t_k. This baseline value E_{h_μ} is then combined with spatially correlated random variations over the area of the sensor network. The harvested energy at sensor i in a given time period t_k, $E_{h_i}(t_k)$ is given by:

$$E_{h_i}(t_k) = E_\mu(t_k) \cdot (1 + Y_i) \quad (3)$$

where $E_{h_\mu}(t_k)$ is the average harvested energy of the entire sensor network during period t_k, and Y_i is the perturbation of the random field at the location of sensor i. Y is assumed to be a zero mean, stationary, gaussian random field in this paper, with standard exponential covariance function:

$$C(x, y) = \sigma^2 \exp\left(\frac{\|x - y\|}{l_0}\right) \quad (4)$$

Realisations of $Y = \left[Y(x_1), ..., Y(x_n)\right]^T$ can then be generated using the Cholesky decomposition method, as described in Dietrich & Newsam (1997).

Given the time series of solar irradiance and meteorological data obtained from the National Solar Radiation Database Habte, Sengupta, & Lopez (2017), and the characteristics of the solar panel, we use the System Advisor Model (SAM) Blair et al. (2014), to calculate the total energy produced for a 2W solar panel, at a 0° tilt, located at $42.3587°N, 71.0915°W$ for every 30 minute interval from January 1st 2008 to December 31st 2013, and draw samples from this to create the baseline harvested energy time series $E_{h_\mu}(t)$.

The addition of spatially correlated noise to this baseline time series ensures that the simulation of solar energy harvesting is more physically realistic: Some regions of the sensor network experience may have lower solar energy availability due to shading or other environmental factors. The stochastic nature of this simulation also provides the reinforcement learning agent with noisy sequences which may help to prevent overfitting and improve robustness to variations in the solar energy availability.

4.3 | Wireless connectivity

Local peer to peer messages are not guaranteed to reach their target, and a method of modelling this uncertainity is needed for a realistic simulation. The soft geometric graph model as

described in Waxman (1988) and Penrose et al. (2016) provides a realistic model of network connectivity, where the probability of connection between node, i and j, is:

$$H_{ij} = \beta \exp\left(-\left(\frac{r_{ij}}{r_0}\right)^{\eta}\right) \tag{5}$$

where r_{ij} is the Euclidean distance between node i and node j, r_0 is a range parameter, η is an environment specific parameter describing signal decay over distance, here assumed to be 1, and β is a distance independent parameter,here also assumed to be 1, which linearly scales the probability of a connection.

The parameter η may vary spatially, if some areas of the sensor network experience higher signal decay than others. For simplicity we assume η has a constant value of 1 over the area of the sensor network.

5 | REINFORCEMENT LEARNING

Reinforcement learning is a framework in which an agent learns to maximise long term utility by interacting with an environment, and receiving rewards. The reinforcement learning problem is based on the Markov Decision Process (MDP) (Bellman, 1957), which is canonically described by a set of possible states $\mathbf{S}$, and actions $\mathbf{A}$, a reward model $R_a(s, s')$, which gives the reward earned by transitioning from each state s to each state s' and a transition model $P_a(s, s')$ which describes the probability of transitioning from state s to state s' if action a is taken. If a sequential decision making problem can be modelled as a MDP, then it is possible to solve for the optimal policy using dynamic programming. The optimal policy defines which action a to take in each state s, with optimality typically defined as the action which maximises the expected value of (possible discounted) future rewards from the current state.

However, in many real world problems, either the transition function is not known explicitly, or the state-action space is too large, causing the solution of the MDP to become intractable. Reinforcement learning allows an agent to learn policies by acting and observing, without an explicit model of the environment. Specifically, in discrete time problems, the RL agent may observe its state at the beginning of each time period, and choose from the set of available actions. At the beginning of the next time period, the agent observes its new state and the reward earned in the previous period, and must then choose its next action. In Q-Learning Watkins (1989), a table of Q-values is maintained for each state-action pair and updated using the Bellman equation:

$$\begin{gathered}\mathbf{Q}_{i+1}(s_i, a_i) = (1-\alpha)\mathbf{Q}_i(s_i, a_i) + \alpha\left(r_i + \gamma \mathbf{V}_i(s_{i+1})\right) \\ \text{where } \mathbf{V}_i(s) = \max_a \mathbf{Q}_i(s, a)\end{gathered} \tag{6}$$

These Q-values are an estimate of the expected discounted reward for executing action a in state s. The Q-learning algorithm is guaranteed to converge to the optimal policy, given an infinite number of iterations. However, when the state-action space is large, the convergence rate may be unacceptably slow, and the Q-table may become too large to practically store. In cases where the state-action space is continuous, quantization can be used, but this too can prove impractical unless the quantization is coarse. Instead of maintaining a lookup table of Q-values, a function approximator can be used to combat these issues. One such approach which has recently shown impressive results in a variety of challenging learning problems is Deep-Q Learning Mnih et al. (2015), which uses a deep neural network to map input state vectors to an estimated Q-value for each possible action.

5.1 | Deep Q Learning

As described in Mnih et al. (2015), the procedure to learn the parameters of the deep Q model θ, involves the minimisation of a loss function at each step of the iteration, to fit the current predicted action-value $Q(s, a; \theta_i)$ to a target y_i. The target y_i is calculated as follows:

$$y_i = r_i + \gamma max_{a'} \mathbf{Q}(s, a'; \theta_{i-1}) \tag{7}$$

Unlike in Mnih et al. (2015), where the mean squared error is used as the loss function, here we use the Huber loss, which is more robust to outliers:

$$\begin{gathered}L(y_i, \mathbf{Q}(s, a; \theta_i)) = L_\delta(y, f(x)) = \\ \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } \|y - f(x)\| \le \delta \\ \delta(\|y - f(x)\|) - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}\end{gathered} \tag{8}$$

The minimisation of $L(\theta_i)$ can be computed by simple gradient descent, or by other stochastic optimisation algorithms. We use the Adam optimisation procedure Kingma & Ba (2014). The optimization can exhibit instability, as both the target and predicted values depend on the model parameters, potentially leading to a positive feedback loop. Note that in Equation 7, the target value y_i is computed using the model parameters θ_{i-1} to avoid this instability. This is achieved by maintaining a target model in addition to the Q-model, which maintains an out of date copy of the model parameters. While Equation 7 suggests the use of the model parameters from the preceding iteration, in practice the target model may be updated less frequently.

Even with the use of a target model, it has been observed that the Deep Q approach tends to overestimate action-values. This tendency can be addressed through a simple adjustment to the target value calculation, as proposed in Van Hasselt, Guez, & Silver (2016), which leads to the Double Deep Q Network (DDQN) formulation. Here, in the evaluation of the target value, the Q-network is used to select the policy, and the target

network is used to determine the value of this policy:

$$y_i^{DDQN} = r_i + \gamma \mathbf{Q}(s, a*, \theta_{i-1}) \quad \text{where} \quad a* =_a Q(s', a; \theta_i) \tag{9}$$

Rather than training the DDQN agent online, we use the experience replay technique, as described in Mnih et al. (2013). At each time step, the agent takes an action a_t which causes a transition from state s_t to state s_{t+1}, and a reward r_t. In online Deep-Q learning, this experience $e_t = (s_t, a_t, r_t, s_{t+1})$, is used to update the model at each step. Instead, in experience replay, the experience e_t is placed in a replay memory buffer. Then random samples of experiences are drawn from the memory buffer to train the model. This has two benefits: 1) Sample efficiency is increased, as each experience can be used more than once in training. 2) Randomising the order of experiences can help to stabilise learning and avoid local minima.

5.2 | Problem formulation

In this section we will describe the behaviour of the system as described in Section 4 in terms of a reinforcement learning problem, and provide details of the experimental procedure used to learn collaborative duty cycling strategies using this model.

The state space representation of node i during time period t_k is given by:

$$S_i(t_k) = [B_i(t_k), X_i(t_k), D_i(t_k)] \tag{10}$$

$X_i(t_k) \in \{0, 1, 2\}$ is an indicator variable that describes whether node i is active, idle or in deep sleep. $B_i(t_k) \in [0, B_{max}]$ can be calculated as in Equation 2, given the consumed energy $E_{c_i}(t_{k-1})$ which depends only on $X_i(t_{k-1})$, as per Equation 1, and the harvested energy in the previous time step $E_{h_i}(t_{k-1})$. $D_i(t_k) \in [-B_{max}, B_{max}]$ is simply the difference in battery between the current and previous time step, i.e. $D_i(t_k) = B_i(t_k) - B_i(t_{k-1})$

The overall system state representation is obtained by concatenating the individual node states into a single vector, and adding one additional state parameter $T_k \in \{0, 1, ..., 24/\Delta_t\}$ which describes the current time of day, corresponding with the k^{th} time step from the initial hour T_0. Concatenating all node states gives the full system representation

$$s(t_k) = \left[B_0(t_k), X_0(t_k), D_0(t_k), B_1(t_k), ..., D_N(t_k), T_k\right] \quad \text{where} \quad T_k = \left(k + T_0/\Delta_T\right) \bmod 24/\Delta_t \tag{11}$$

The order of entries into the state vector $s(t_k$ is determined using the Hilbert space-filling curve Moon, Jagadish, Faloutsos, & Saltz (2001). The Hilbert curve provides a mapping from 2D coordinates to 1D space, which we use to sort the nodes in the network. This has the advantage of ensuring that nodes which are adjacent in the state vector are also physically close in 2D space. The converse is not true; some coordinates which are close in 2D space are not close on the Hilbert curve.

For each node, there are two available actions, a_{i_0}, and a_{i_1}. a_{i_0} toggles the idle behaviour of node i: If the node is idle, it becomes active, otherwise it switches to idle. a_{i_1}, similarly toggles the deep sleep behaviour of node i, causing it to switch to deep sleep from either idle or active, and to active if it is in deep sleep. Only one of these actions, or the null action ∞ can be executed in any time step, so the full action space is given by $\mathbf{A} = \{a_{0_0}, a_{0_1}, a_{1_0} ... a_{N_0}, a_{N_1}, \infty\}$

5.2.1 | Reward function

Given the system state representation for a time period s_{t_k} we can cut the set of nodes in the network, G, into the set of active nodes $\mathrm{C_a}(t_k)$, the set of idle nodes $\mathrm{C_i}(t_k)$, the set of nodes in deep sleep $\mathrm{C_s}(t_k)$, and the set of nodes which are out of battery $\mathrm{C_0}(t_k)$. Active nodes, for example, have non-zero battery life $B_i > 0$ and have active status $X_i = 2$:

$$\mathbf{C_a}(t_k) = \{i \in \mathrm{G} \mid X_i(t_k) = 2 \;\&\; B_i(t_k) > 0\} \tag{12}$$

The soft geometric graph model described in Section 4 defines the probability of a successful connection between node i and node j during a given time period t_k. Using Equation 5, the set of nodes reached by node i, $\mathrm{R_i}$ is easily obtained by Monte Carlo simulation:

$$\mathbf{R_i}(t_k) = \{j \in \mathrm{C_a(t_k)} \cup \mathrm{C_i(t_k)} \mid H_{ij} > x \sim \mathcal{U}(0, 1)\} \tag{13}$$

Note here that we do not include nodes in deep sleep mode, or nodes with zero battery in the set of reachable nodes. Combining Equations 13 and 12, the set of all reachable nodes in a given time period is the union of the nodes reachable by each active node:

$$\mathrm{R}(t_k) = \bigcup_{i \in C_a} \mathrm{R_i} \tag{14}$$

Finally, we can define the reward earned in period t_k given the system state s_{t_k} as the number of reachable nodes divided by the total number of nodes.

$$r(s_{t_k}) = \begin{cases} \frac{|R(t_k)|}{|G|} & \text{when } |R(t_k)| > 0 \\ -1 & \text{when } |R(t_k)| = 0 \end{cases} \tag{15}$$

Note that a penalty is accrued if no nodes are active during a given time period. This is designed to heavily disincentivise this outcome, as missing the occurrence of an event completely is highly undesirable: Having just one node on for a time period is much preferable to having none, whereas having two instead of one is a relatively less important distinction.

5.3 | Experimental procedure

In this section we will describe in detail the procedure used to emulate the sensor network system behaviour, and train a

Algorithm 1 DDQN adaptive duty cycling learner

Initialise:
Replay memory buffer D.
Master baseline solar energy series E_{h_μ}.
Exploration variable ϵ=1, $\epsilon_{min} = 0.01$.
Q network with random weights θ and duplicate target network.

for episode = 1 to M **do**
$\{(x_i, y_i)\ \forall i \in \mathbf{G}\} \leftarrow$ Generate node coordinates.
$T_0 \leftarrow$ Initial time stamp.
$t_{episode} \leftarrow [T_0, T_0 + \Delta_T, ..., T_0 + N\Delta_T]$.
$Y \leftarrow$ new random field realisation.
$E_{h_i}(t_{episode}) \leftarrow E_{h_\mu}(t_{episode}) \cdot (1 + Y_i)$.
$s_0 \leftarrow$ Initial state.
for k = 1 to N **do**
If $\epsilon > x \sim \mathcal{U}(0, 1)$ select action a_t randomly, otherwise select $a_t = \max_a Q(s_t, a; \theta)$.
Observe resulting s_{t+1} and calculate $r(t)$.
Push $e_t = (s_t, a_t, r_t, s_{t+1})$ to D.
Set $s_t \leftarrow s_{t+1}$.
Sample minibatch of experiences e_j from D.
Set y_j for e_j as per Equation 9 and perform optimization step to minimize $L(y_j, Q(s, a_j; \theta_j)$ as per Equation 8.
Set $\epsilon = \epsilon - (\epsilon_0 - \epsilon_{min})/NM$.
end for
Copy weights θ from Q to target network.
end for

Double Deep Q Learning agent on this emulation. Algorithm 1 provides a step by step description of the simulation and training.

5.3.1 | Global initialisation

To begin the training procedure, the replay memory buffer is first set, with a fixed length of 2000, and the Q-network and target network are initialised with identical, randomly generated weights. Then the master time series of baseline solar energy is calculated, as detailed in Section 4.2, which will be sampled from in each training episode. The exploration variable ϵ controls the degree of randomness the agent exhibits in selecting actions. If this value is too high, the agent explores different strategies well, but fails to focus on and fine tune the most promising strategies. Conversely, if this value is too low, the agent quickly hones on the most immediately rewarding strategy, but may fail to explore other more valuable strategies. Here we start with an ϵ of 1, and linearly decrease it so it reaches a minimum value of 0.01 on the last time step of the last episode.

5.3.2 | Episode initialisation

At the beginning of each new episode, the spatial coordinates of each node $i \in \mathbf{G}$ are generated. The configuration of the nodes is arbitrary, as the methodology presented in this paper is valid for any configuration in 2D or 3D space. Furthermore, the spatial configuration can be changed from episode to episode in an attempt to learn non configuration-specific policies. For simplicity, in this paper we select a 2D square lattice of 16 nodes as shown in Figure 2 , and keep this configuration for the entire simulation.

Each training episode consists of a number of time steps N. At the beginning of each episode, a start time T_0 is chosen. The master solar energy time series is then sampled for the period defined by the series $t_{episode}$, creating the specific baseline solar energy time series for that episode. Given the node coordinates, an instance of the random field $\mathbf{Y}$ is then created for the current episode, which allows for the calculation of the harvested solar energy at each node $E_{h_i}(t_{episode})$ as per Equation 3. Figure 2 , illustrates an instance of the random field over the lattice network. Values of the random field over the entire space of the 2D grid are shown only for illustrative purposes; it is only necessary to simulate realisations of this random field at the specified node locations. The results presented in this paper are generated with a discretisation parameter $\Delta_T = 3hrs$, and with T_0 fixed at September 1st at midnight for every episode, and $N = 30 \cdot \frac{24}{\Delta_T} = 240$, such that $t_{episode}$ represents the entire month of September in each case. Although T_0 is fixed at September 1st at midnight in every episode, the chosen year is varied randomly in every episode, so a representative variety of conditions are experienced by the reinforcement learning agent. 2013 is excluded for testing purposes.

Shown in Figure 3 are the time series of harvested energies (in Watts) at each node in the lattice network for two different days in September 2012. Note here the significant variability in available energy between the two days, most likely due to the contrast between overcast and sunny conditions. Note also the variance between individual nodes arising from the random field perturbation. The exponential covariance function used for random field simulation is that given by Equation 4, with $\sigma^2 = 0.1$, and $l_0 = l$ where l is the total width of the lattice network. σ^2 controls the variance of the random field, while l_0 controls the length scale of the spatial correlation between nodes. We can see that these random field parameters result in a small but noticeable deviation in individual node solar energy as compared to the baseline , where all nodes harvest equal energy (i.e. $\sigma^2 = 0$).

Once all episode level initialisation is complete, the inner training loop begins. At each time step in the episode,an action is selected randomly, with probability ϵ, which is very high in early episodes. Otherwise the action which the Q-network deems to have the highest value is selected. This action is then

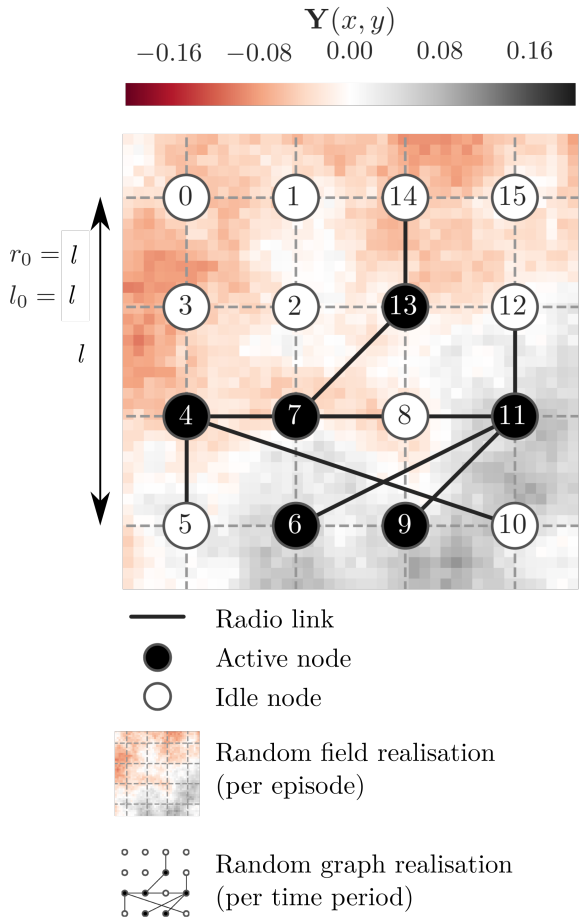

FIGURE 2 Illustration of chosen lattice network configuration, with a typical random field instance, active and idle nodes assignment, and instantaneous network connectivity.

executed, resulting in a transition to a new state, and an earned reward. The experience corresponding with this iteration of the episode is then pushed to the replay buffer. If the buffer is full, the new experience will overwrite the oldest experience in the buffer. A minibatch is sampled from the replay buffer, and the Q network is updated. Finally, the exploration variable ϵ is decremented, causing the agent to very gradually prefer exploitation rather than exploration. Once the episode ends, the target model is updated with a copy of the Q-networks weights, before a new episode is initialised.

5.3.3 | Network architecture

Figure 4 shows the neural network architecture used for both the Q network and the target network. Our lattice configuration of 16 nodes, each with 3 state variables (battery, status, battery difference) as well as the global time variable, gives an input vector of length 49. This input is then fed through 3 identical fully connected layers of 512 neurons, each with a rectified

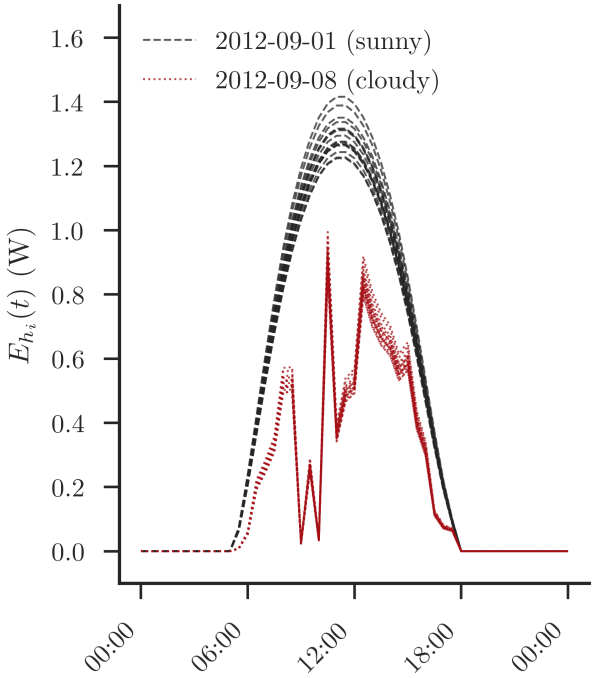

FIGURE 3 Harvested energy by node for two contrasting days.

linear unit (ReLu) activation function. Finally, the output layer consists of 33 neurons, which predict the Q-value of each of the available actions. The network is constructed and trained using the Keras library Chollet et al. (2018).

6 | RESULTS

In this section we describe the results obtained by training our DDQN agent for 6500 episodes on the lattice network configuration. For benchmarking purposes, we will first define two baseline agents. First, the *eno* agent, which optimises the duty cycle level of each individual node in the network, using the linear programming approach described in Kansal et al. (2006), and assuming perfect knowledge of the future solar energy time series. This agent sets a percentage duty cycle level for each time step, and maximises the sum of these duty cycle levels, while ensuring that the battery level never falls below zero, and that the battery level at the end of each day is higher than or equal to the previous day. Nodes are in active mode for the given duty cycle percentage of each time step, and otherwise in deep sleep mode. Additionally, it is assumed that within a given time period, the probability of being active is equally likely at all times. This implies that each individual node behaves independently of all others. Secondly, we define

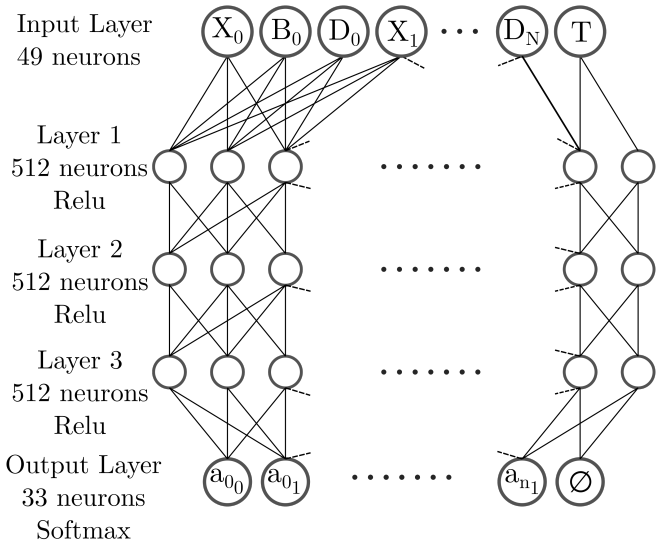

FIGURE 4 Neural network architecture for both Q and target networks

a *greedy* agent, closely related to the LEACH approach for adaptive clustering proposed in Heinzelman et al. (2002). The greedy agent aims to keep 50 percent of the nodes in the network active at every time step. If fewer than half the nodes are active, it greedily adds the idle node with the maximum battery life. Additionally at every time step, when possible, the *greedy* agent churns active nodes with low battery in favour of any available idle nodes with more battery life. More formally, the policy of the greedy agent can be stated as in Algorithm 2:

Algorithm 2 Greedy adaptive duty cycling agent

for k = 1 to N **do**
  Find active set $\mathbf{C_a} \subseteq \mathbf{G}$ as per Equation 12
  Find idle set $\mathbf{C_s} = \mathbf{G} - \mathbf{C_a}$
  Find idle node $j = \max_{j \in C_s} B(j) > 0$ with maximum battery life
  Find active node $i = \min_{i \in C_a} B(i)$ with minimum battery life
  If $|\mathbf{C_a}| \leq |\mathbf{G}|/2$ then take action a_j
  Elif $B(i) < B(j)$ then take action a_i
  Else take action ∞
end for

Shown in Figure 5 , is the learning curve of the DDQN agent, compared with both the *greedy* and *eno* agents. Here we define the performance metric of interest as the episode score, or the sum of rewards, as defined in Equation 15, earned over the course of the episode. The initial performance of the DDQN agent is very poor, as it acts almost entirely randomly in early episodes while it explores the state space. The performance of the *greedy* agent is significantly better than the *eno* agent, primarily because the *eno* agent frequently accrues negative penalties in night-time periods when all nodes enter a 0 %duty cycle. As training progresses, the DDQN agent improves its performance, eventually surpassing both the *eno* agent and the *greedy* agent.

Figure 6 compares the performance of all three agents on both the held out unseen data, and data included in the training set. Shown in this figure is the results of 30 individual runs for each agent on both sets of data, with the mean, and the interquartile range indicated by the box plot. While the performance of the DDQN agent is slightly worse on the unseen data, this appears to be the case for both of the baseline agents also, indicating that this is due to a worse energy harvesting environment, rather than overfitting.

In Figure 7 , we illustrate the difference in earned rewards for each agent over every 3 hour period in a full week of unseen data, to investigate where the DDQN agent gains an advantage over the baseline approaches. The DDQN agent and the greedy agent exhibit similar performance for many of the time periods. However, the DDQN agent appears more robust to multi-day periods of low solar energy availability (e.g. September 3rd and September 4th), and therefore avoids entering a state where no nodes are active. This further demonstrates that the DDQN agent's robustness to variations in solar energy, as it

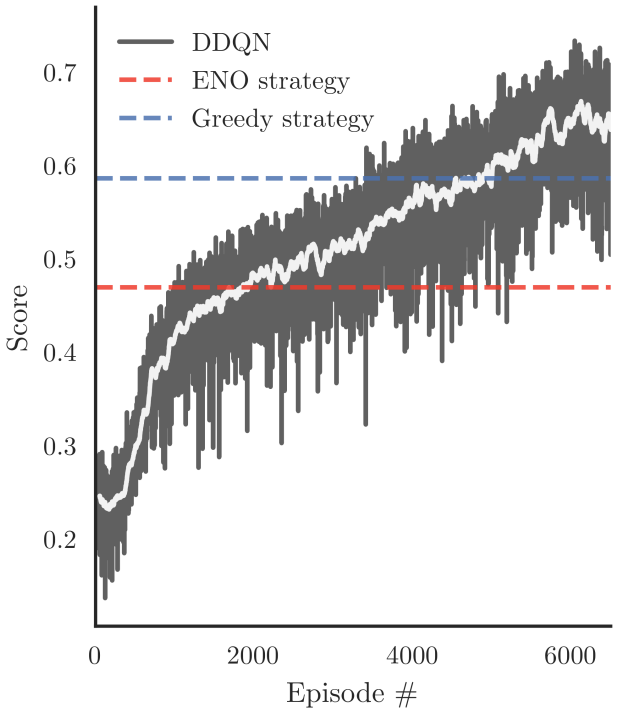

FIGURE 5 Learning curve of the DDQN agent, compared to random and greedy agents.

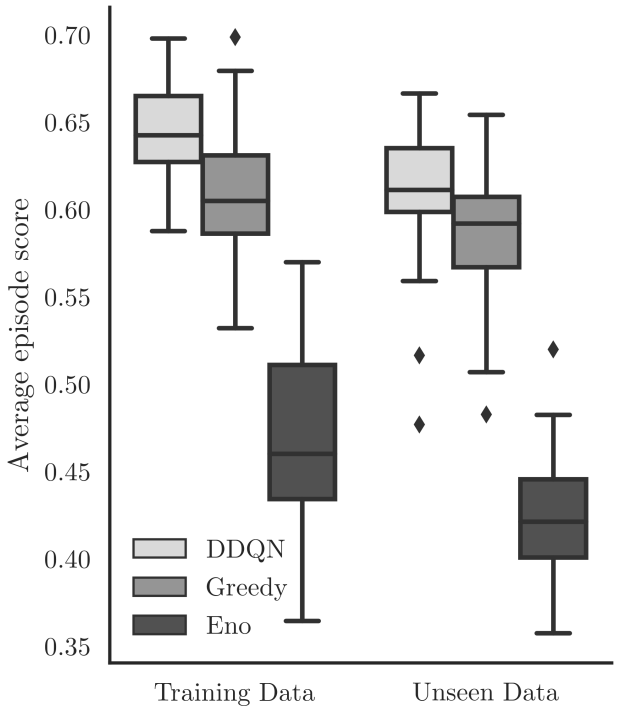

FIGURE 6 Performance of the DDQN agent compared with greedy and eno agents on both unseen and training data

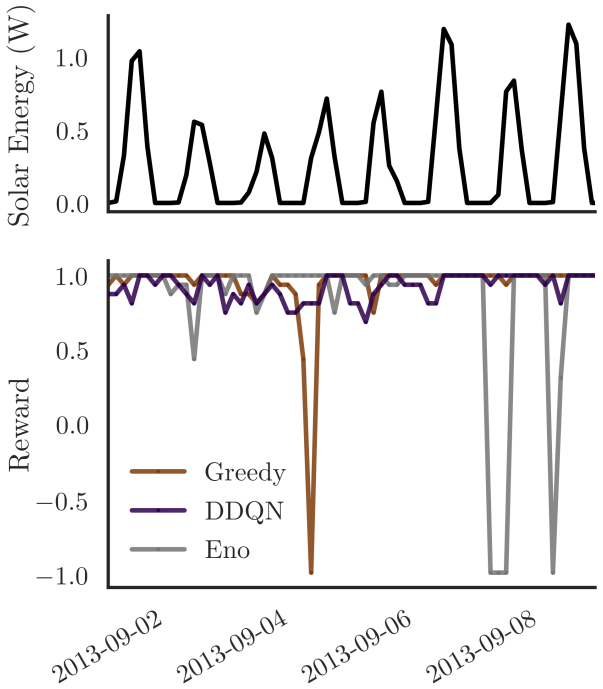

FIGURE 7 Comparison of the earned rewards of each agent over a week long period.

demonstrates this capability on a sequence of solar energy data which it has not previously been trained on. Unsurprisingly, the ENO agent, which explicitly optimizes to avoid using more energy than it harvests over a 24 hour period, is more susceptible to losing event detection capability at night time when no solar energy is available.

Further analysis of the behaviour of the system under the control of the trained agent is necessary to understand how the learned policy outperforms the baselines. For a given time step, we define the configuration of the sensor network to be the vector containing just the status variable of each node $C(t_i) = [X_0(t_i), X_1(t_i), ...X_N(t_i)]$. Then for each unique configuration, we count the transition frequencies to each other unique configuration, to estimate an empirical transition matrix P. If n_{jk} is the number of occurrences where $C(t_i) = j \quad \& \quad C(t_{i+1}) = k$, then assuming there are m total unique configurations, each element of the transition matrix can be estimated as $P_{ij} = n_{ij} / \sum_{k=1}^{m} n_{ik}$. Provided this matrix corresponds with an irreducible and aperiodic Markov chain, we can analyse its steady state behaviour. The steady state distribution of a Markov chain represents the long term probability of being in any given state, independent of the initial conditions. This provides a useful tool for interpreting patterns in simulated time series to understand the strategy of the trained reinforcement learning agent. Given the empirical transition matrix P, the steady state distribution π can be found by solving the eigenvalue problem $(\mathrm{P} - \lambda \mathrm{I})\pi = 0$, for $\lambda = 1$. The 6 configurations with the highest steady state probabilities are shown in Figure 8 .

Inspecting Figure 8 , we see several different behaviours of the network emerge. Two of these configurations, 1) and 5), show node 2 as the only active node, while the remaining nodes are either idle or out of battery. Two more of the configurations ,3) and 4), show 50 % of the nodes active, with the remainder of the nodes idle, with the exception of node 2 which is in standby mode.

Figure 9 yields further insight into the strategy of the DDQN agent. In this figure a 5 day slice is illustrated, with each node's behaviour indicated by a vertical line: Solid black indicates the node is active, dashed black indicates idle, dashed red indicated standby, and a thin dotted line indicates the node is out of battery. Here we can clearly see the cyclical behaviour of the network. Active nodes tend to run out of battery at night time when solar energy is not available. In order to maintain the networks event detection capability during these periods, node 2 enters standby mode in advance, allowing it to to preserve battery and then assume active duty while other nodes have depleted their reserves.

This behaviour may raise questions of reliability: In a scenario where only one node is on active duty, how robust

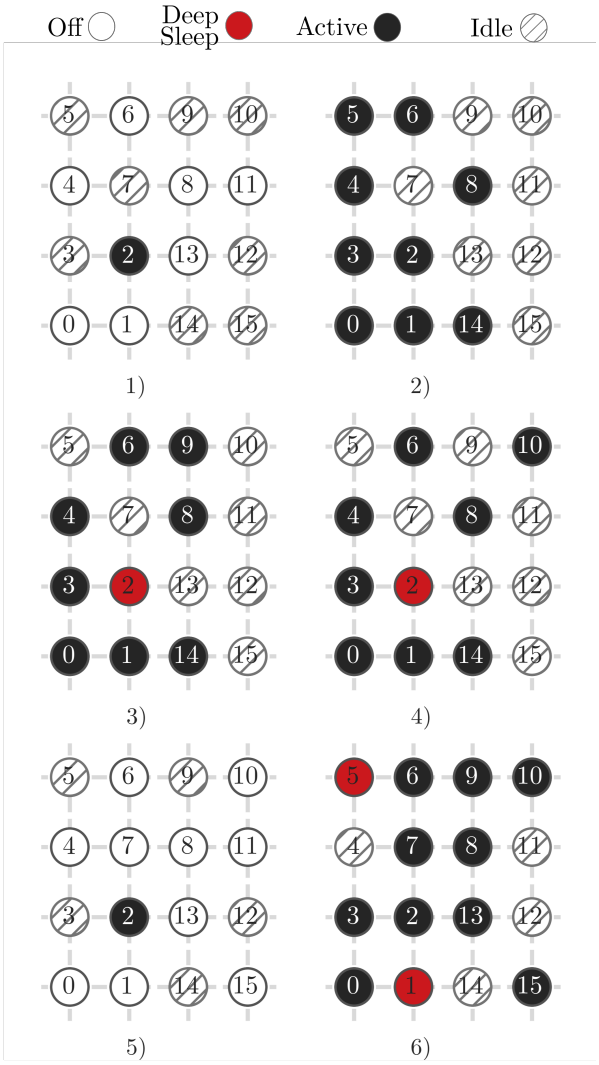

FIGURE 8 Highest probability steady state configurations of the 16 node lattice network under the control of the trained DDQN agent.

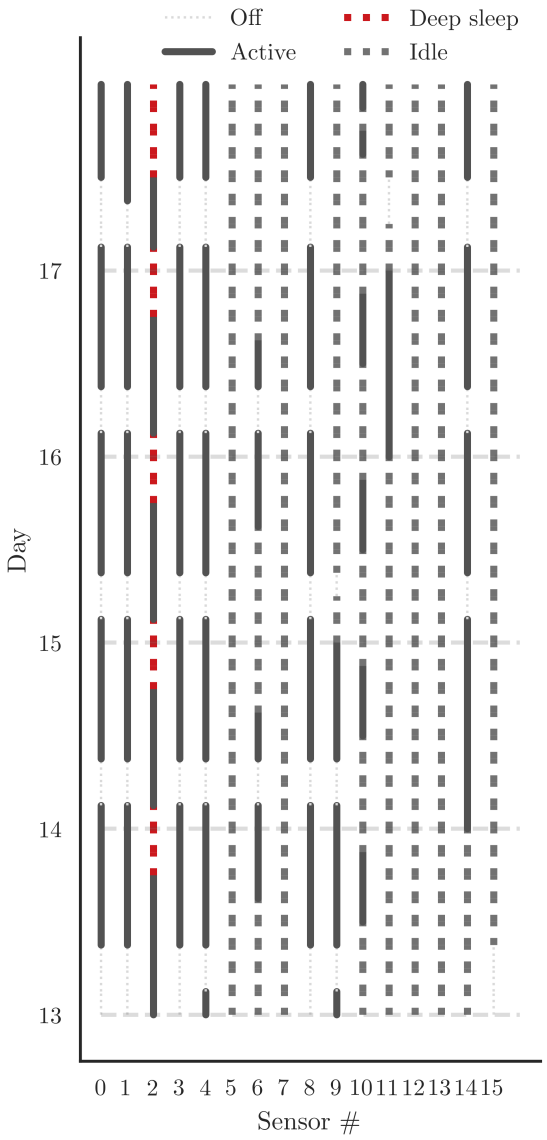

FIGURE 9 Detailed evolution of network wide statuses over time

is the system to communication failures? Although this is a valid concern, the simulation environment does incorporate a stochastic simulation of communication success, which is reflected in the reward function. Although the trained reinforcement learning agent earns a higher reward than other approaches on average, there may be a concern that its strategy causes a higher variance: However, examining Figure 6 , we see that the variance of the average episode score for the reinforcement learning agent is similar to the other approaches.

An alternative view of the behaviour of the newtork under the control of the trained agent is shown in Figure 10 , where the spatio-temporal behaviour of the network over a full day cycle is illustrated. This figure shows the evolution of both status and battery reserve over time, with battery reserve indicated by the arc shown around each node location. Investigating the spatial behaviour of the DDQN strategy we can see a few interesting patterns. Examining the diagonal set of nodes, 5,7,13 and 15, we can see that in this time slice, these nodes tend to remain idle, with active nodes off this diagonal. Finally, we investigate the behaviour of the network over a longer time period, as shown in Figure 11 . Here we select 5 nodes which exhibit distinct groups of long term behaviours. In this figure, we examine the ratio of time active to time idle for each node, averaged over a single day. Note that for clarity, we omit time

periods when nodes are out of battery. Node 0 spends 100 % of its on time in active duty. In comparison node 13 spends almost 100 % of its time in idle duty. Other nodes switch between these opposing behaviours, but generally exhibit either one or the other over the course of a full day. For example node 4 spends almost 100 % of its on time in idle mode in the first 10 days, but then switches to spend the entire remaining days in active duty only. Node 11 spends long periods in idle mode, but intermittently switches to spend an entire day or two in active duty. Finally, node 2, as we have already seen, exhibits an entirely different behaviour: It is active much more often than it is idle, but never spends a full day in active duty. It does not spend any time in idle mode, but rather in standby mode, as shown in Figure 10 , and Figure 9 , allowing it to preserve battery.

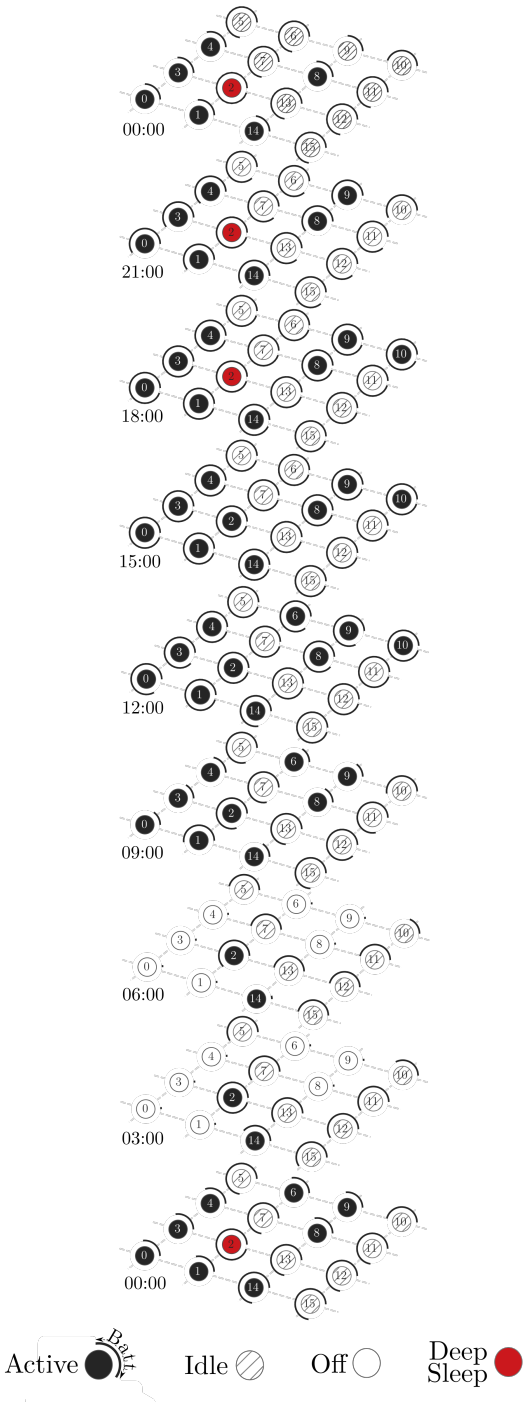

FIGURE 10 Spatio-temporal visualisation of evolution of network statuses and battery levels over a full day.

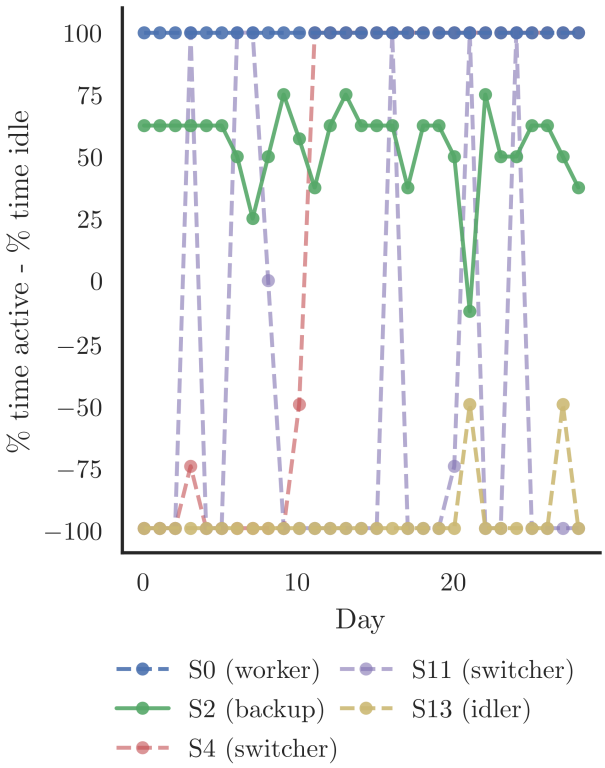

FIGURE 11 Long term evolution of nodes behaviour, averaged over each day.

7 | CONCLUSIONS

In this study a novel reinforcement learning based approach for adaptive duty cycling in energy harvesting sensor networks is proposed. This approach considers specifically the system requirements for event detection applications. Within this context, a system wide performance metric is developed, and a reinforcement learning agent is formulated to maximise this performance metric by controlling node duty cycling.

To validate this approach a simulation environment is developed which emulates node power consumption, energy harvesting and connectivity properties. We consider specifically a 16 node network in a 2D lattice configuration, where each node is equipped with a small photovoltaic panel. Using this simulation environment, the proposed reinforcement learning approach is tested against two different baseline strategies, and outperforms each on both training data and unseen data

Investigation of the learned agents behaviour yields several insights. Most notably, the advantage of the sequential decision making framework used in this paper is demonstrated clearly: Some nodes exhibit foresight, preserving battery in times of solar energy availability, in order to later assume active duty when no energy is available. This can be viewed as a sort of 'rainy day' planning. When compared with the baseline agents, the DDQN agent appears as a result to demonstrate more robustness to periods of low solar availability, and as a result, can avoid scenarios in which no node is available for active duty. Secondly, the agent also learns what proportion of nodes to maintain in active duty when battery is available: Even though all nodes have battery reserves, only a subset of these nodes need to be in active duty to reach the entire network.

7.1 | Future work

In this work we make the simplifying assumption that all node locations are equally capable of detecting events. Considering varying detection capabilities is an interesting avenue for future work, especially in monitoring applications where events tend to propagate in an obvious pattern. Consider a bridge monitoring system, where events of interest are triggered by heavy goods vehicle loading. In this instance, nodes located near the ends of the bridge may provide greater utility by allowing earlier event detection.

Relatedly, we assume in this work that data recorded by each node is of equal value. In reality, in vibration analysis applications, the quality of the estimated dynamic properties will depend on the set of nodes chosen for recording. One possible future extension of the framework presented in this paper would be to modify the reward function to reflect the expected information value of the reachable nodes.

In addition, here we assume that events of interest are equally likely to occur at any time. In many applications, event occurrence is non-random. Consider again a bridge monitoring application: Traffic loading shows distinct temporal peaks. By incorporating a model of event arrival into the reward function, our reinforcement learning agent could learn that it is more important to maximise coverage at certain times.

Although the reinforcement learning approach proposed in this paper exhibits good performance once trained, training is relatively slow. Recent work on reinforcement learning from demonstrations (Hester et al., 2017), allows an agent to learn from its own exploration, combined with system behaviour under the control of an expert. Because we have access to relatively sophisticated human created strategies, such as the greedy agent described in this paper, this technique could help to accelerate the agent's learning curve.

Finally, to validate the capability and advantages of the proposed approach, a case study demonstrating the performance of a deployed sensor network in a real structural monitoring application would be of interest. The reinforcement learning based strategies developed in this paper are learned from interaction with a simulated environment. In a real deployment, there may be inconsistencies between the simulation environment, and the real physical environment, which decrease the efficacy of the learned strategy. One potential avenue to account for this issue is to use techniques from transfer learning Taylor & Stone (2009), to efficiently fine tune the performance of the reinforcement learning agent using observed data from the real deployment.

References

Addabbo, T., Fort, A., Mugnaini, M., Panzardi, E., Pozzebon, A., & Vignoli, V. (2019). A city-scale iot architecture for monumental structures monitoring. *Measurement*, *131*, 349–357.

Bahbahani, M. S., & Alsusa, E. (2017). Dc-leach: A duty-cycle based clustering protocol for energy harvesting wsns. In *Wireless communications and mobile computing conference (iwcmc), 2017 13th international* (pp. 974–979).

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, 679–684.

Blair, D. A. P., Nate, Freeman, J., Neises, T., Wagner, M., Ferguson, T., Gilman, P., & Janzou, S. (2014). System advisor model, sam 2014.1. 14: General description. report nrel/tp-6a20-61019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Chan, W. H. R., Zhang, P., Nevat, I., Nagarajan, S. G., Valera, A. C., Tan, H.-X., & Gautam, N. (2015). Adaptive duty cycling in sensor networks with energy harvesting using continuous-time markov chain and fluid models. *IEEE Journal on Selected Areas in Communications*, *33*(12), 2687–2700.

Chollet, F., et al. (2018). Keras: The python deep learning library. *Astrophysics Source Code Library*.

Dietrich, C. R., & Newsam, G. N. (1997). Fast and exact

simulation of stationary gaussian processes through circulant embedding of the covariance matrix. *SIAM Journal on Scientific Computing*, *18*(4), 1088–1107.

Fang, K., Liu, C., & Teng, J. (2018). Cluster-based optimal wireless sensor deployment for structural health monitoring. *Structural Health Monitoring*, *17*(2), 266–278.

Fu, T. S., Ghosh, A., Johnson, E. A., & Krishnamachari, B. (2013). Energy-efficient deployment strategies in structural health monitoring using wireless sensor networks. *Structural Control and Health Monitoring*, *20*(6), 971–986.

Gao, Y., Spencer Jr, B. F., & Ruiz-Sandoval, M. (2006). Distributed computing strategy for structural health monitoring. *Structural Control and Health Monitoring*, *13*(1), 488–507.

Habte, A., Sengupta, M., & Lopez, A. (2017). *Evaluation of the national solar radiation database (nsrdb): 1998-2015* (Tech. Rep.). National Renewable Energy Lab.(NREL), Golden, CO (United States).

Heinzelman, W. B., Chandrakasan, A. P., & Balakrishnan, H. (2002). An application-specific protocol architecture for wireless microsensor networks. *IEEE Transactions on wireless communications*, *1*(4), 660–670.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., … others (2017). Deep q-learning from demonstrations. *arXiv preprint arXiv:1704.03732*.

Hsu, R. C., Liu, C.-T., & Wang, H.-L. (2014). A reinforcement learning-based tod provisioning dynamic power management for sustainable operation of energy harvesting wireless sensor node. *IEEE Transactions on Emerging Topics in Computing*, *2*(2), 181–191.

Huang, S.-C., & Jan, R.-H. (2004). Energy-aware, load balanced routing schemes for sensor networks. In *Parallel and distributed systems, 2004. icpads 2004. proceedings. tenth international conference on* (pp. 419–425).

Huang, Y., Beck, J. L., & Li, H. (2018). Multitask sparse bayesian learning with applications in structural health monitoring. *Computer-Aided Civil and Infrastructure Engineering*.

Jang, S., Jo, H., Cho, S., Mechitov, K., Rice, J. A., Sim, S.-H., … Agha, G. (2010). Structural health monitoring of a cable-stayed bridge using smart sensor technology: deployment and evaluation. *Smart Structures and Systems*, *6*(5-6), 439–459.

Kansal, A., Hsu, J., Srivastava, M., & Raghunathan, V. (2006). Harvesting aware power management for sensor networks. In *Proceedings of the 43rd annual design automation conference* (pp. 651–656).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Liu, X., Cao, J., Tang, S., & Wen, J. (2016). Enabling reliable and network-wide wakeup in wireless sensor networks. *IEEE Transactions on Wireless Communications*, *15*(3), 2262–2275.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … others (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Moon, B., Jagadish, H. V., Faloutsos, C., & Saltz, J. H. (2001). Analysis of the clustering properties of the hilbert space-filling curve. *IEEE Transactions on knowledge and data engineering*, *13*(1), 124–141.

Penrose, M. D., et al. (2016). Connectivity of soft random geometric graphs. *The Annals of Applied Probability*, *26*(2), 986–1028.

Peters, J., & Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, *21*(4), 682–697.

Rafiei, M. H., & Adeli, H. (2017). A novel machine learning-based algorithm to detect damage in high-rise building structures. *The Structural Design of Tall and Special Buildings*, *26*(18), e1400.

Rafiei, M. H., & Adeli, H. (2018). A novel unsupervised deep learning model for global and local health condition assessment of structures. *Engineering Structures*, *156*, 598–607.

Rice, J. A., Mechitov, K., Sim, S.-H., Nagayama, T., Jang, S., Kim, R., … Fujino, Y. (2010). Flexible smart sensor framework for autonomous structural health monitoring.

Shah, R. C., & Rabaey, J. M. (2002). Energy aware routing for low energy ad hoc sensor networks. In *Wireless communications and networking conference, 2002. wcnc2002. 2002 ieee* (Vol. 1, pp. 350–355).

Shresthamali, S., Kondo, M., & Nakamura, H. (2017). Adaptive power management in solar energy harvesting sensor node using reinforcement learning. *ACM Transactions on Embedded Computing em ems (TECS)*, *16*(5s), 181.

Sim, S.-H., Carbonell-Márquez, J. F., Spencer Jr, B. F., & Jo, H. (2011). Decentralized random decrement technique for efficient data aggregation and system identification in wireless smart sensor networks. *Probabilistic Engineering Mechanics*, *26*(1), 81–91.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine*

Learning Research, *10*(Jul), 1633–1685.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Aaai* (Vol. 2, p. 5).

Vigorito, C. M., Ganesan, D., & Barto, A. G. (2007). Adaptive control of duty cycling in energy-harvesting wireless sensor networks. In *2007 4th annual ieee communications society conference on sensor, mesh and ad hoc communications and networks* (pp. 21–30).

Watkins, C. J. C. H. (1989). *Learning from delayed rewards* (Unpublished doctoral dissertation). King's College, Cambridge.

Waxman, B. M. (1988). Routing of multipoint connections. *IEEE journal on selected areas in communications*, *6*(9), 1617–1622.

Xiao, M., Zhang, X., & Dong, Y. (2013). An effective routing protocol for energy harvesting wireless sensor networks. In *Wireless communications and networking conference (wcnc), 2013 ieee* (pp. 2080–2084).