Maximizing network throughput by cooperative reinforcement learning in clustered solar-powered wireless sensor networks

Yujia Ge\textsuperscript{1,2} \textsuperscript{\textcopyright}, Yurong Nan\textsuperscript{1} and Xianhai Guo\textsuperscript{3}

Abstract
Power management in wireless sensor networks is very important due to the limited energy of batteries. Sensor nodes with harvesters can extract energy from environmental sources as supplemental energy to break this limitation. In a clustered solar-powered sensor network where nodes in the network are grouped into clusters, data collected by cluster members are sent to their cluster head and finally transmitted to the base station. The goal of the whole network is to maintain an energy neutrality state and to maximize the effective data throughput of the network. This article proposes an adaptive power manager based on cooperative reinforcement learning methods for the solar-powered wireless sensor networks to keep harvested energy more balanced among the whole clustered network. The cooperative strategy of Q-learning and SARSA(\lambda) is applied in this multi-agent environment based on the node residual energy, the predicted harvested energy for the next time slot, and cluster head energy information. The node takes action accordingly to adjust its operating duty cycle. Experiments show that cooperative reinforcement learning methods can achieve the overall goal of maximizing network throughput and cooperative approaches outperform tuned static and non-cooperative approaches in clustered wireless sensor network applications. Experiments also show that the approach is effective in response to changes in the environment, changes in its parameters, and application-level quality of service requirements.

Keywords
Solar-powered wireless sensor networks, Internet of things, cooperative reinforcement learning, adaptive power management, energy harvesting

Date received: 17 February 2021; accepted: 9 March 2021
Handling Editor: Miguel Acevedo

Introduction
A key challenge that limits the applications of wireless sensor networks (WSNs) is the battery lifetime generated by the restricted energy storage of each sensor node. Energy harvesting wireless sensor networks (EH-WSNs) can extract energy from environmental sources as supplemental energy to break this limitation.\textsuperscript{1,2} The latest developments in energy harvesting technology can greatly extend the sensor network lifetime and may

\textsuperscript{1}College of Information Engineering, Zhejiang University of Technology, Hangzhou, China
\textsuperscript{2}School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, China
\textsuperscript{3}Department of Computer Science and Technology, Taizhou University, Taizhou, China

Corresponding author:
Yujia Ge, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China.
Email: yge@mail.zjgsu.edu.cn
run permanently without replacing sensor batteries strenuously which is even not feasible in some cases. The energy that can be harvested often varies with time and uncertainty. Consistent with the node’s energy collection and node workload, the use of adaptive energy management strategies to improve energy utilization can thereby improve the system performance and system sustainable operations. At the same time, the energy harvesting efficiency of energy harvesting batteries is affected by weather, climate, and location. Bandwidth is also an important challenge. These limitations require sensor networks to be able to adapt to the changing environment and effectively use the available energy. Besides, in practical applications, WSNs need to adapt to changes in device parameters and system degradation to dynamically meet application-level quality of service (QoS) requirements (such as network delay, data throughput, and energy consumption).

To solve the above problems, approaches in developing WSN protocols with low power consumption, high performance, and low latency, such as medium access control (MAC) protocol, routing protocols, and data aggregation methods, are developed. The advantage of these solutions is that the system’s low power consumption is achieved, but the nodes are difficult to achieve long-term operation. Even if solar cells are used as energy sources, the system cannot adapt to changes in utilizing the harvested energy as much as possible. The node cannot either adjust automatically the duty cycle and information transmission rate to obtain optimal performance. Due to the dynamic nature of energy harvesting resources in WSNs, as well as the variation of network environment such as different network topology, node residual energy, and energy consumption, the static approach apparently cannot be good enough to solve the problems faced by EH-WSNs.

Reinforcement Learning (RL) is a learning method aiming at maximizing the cumulative reward of system behavior from the environment, which is suitable for adaptive energy management strategies. RL is mainly divided into value-based methods and policy-based methods, as well as the combined methods known as Actor-Critic methods. RL algorithms have been practically used in many fields, and they also have certain applications in energy management in the Internet of things (IoT). Based on the combination of its learning and the interaction of the environment, the agent can react to the changes in the changing environment to improve the performance of WSNs.

A general RL model consists of a finite state space $S$, an agent that can perform a set of actions $A$, and an environment that responds to the actions. The environment reacts to the agent’s action with a reward defined by the reward function $R : S \times A \rightarrow R$. The agent takes one step for each interaction with the environment. In step $t$, when the agent is in state $s \in S$, the agent takes an action $a \in A(s)$ based on the strategy $\pi$ which is expressed in equation (1)

$$\pi = \{(s, a) | a \in A, s \in S\}$$ (1)

The environment’s response to this action is to change the state of the agent to a new state $s' \in S$ and reward the agent with reward $r$. The goal of the agent is to find a strategy to maximize the expectations of the return. An optimal strategy maximizes the value of all state-action pairs. To measure the value of specific action $a$ in a certain state $s$, according to the strategy $\pi$, the value $q_\pi(s, a)$ is defined in equation (2)

$$q_\pi(s, a) = E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$ (2)

where $R_{t+k+1}$ is the reward of the action sequences in the current state, and $\gamma (0 < \gamma < 1)$ is the discount factor. The $Q$-value of the optimal strategy $\pi^*$ is expressed by $q^*$. Once $q^*$ is known, for each state $s$, the action $q_\pi(s, a)$ that maximizes it is the optimal action.

In a cooperative way of RL, an agent interacts with other agents in the same environment. The rewards and actions’ information could be shared in different strategies among these agents. For example, an agent changes from state $s$ to $s'$ after performing a joint action $a_i$, and the reward $R_i(s, a)$ for agent $i$ is then calculated after the joint action $a$ is taken in state $s$. The sum of all individual rewards received by the $n$ agents is finally returned. Different communication mechanism could affect possible solutions in an environment, such as how agents communicate with each other, whether agents share the joint action information, and whether agents share the other agents’ rewards.

This article proposes cooperative reinforcement learning (CRL) methods, specifically $Q$-learning and SARSA($\lambda$), for adaptive energy management of EH-WSNs. CRL methods are used to achieve automatic energy management in IoT nodes in a clustered EH-WSN. The reward function is designed for maximizing the amount of valid data that the sink node finally received during a whole day, and minimizing the distance that the node leaves the energy neutrality state. The adjustable duty cycle sensor nodes of the whole network are grouped into clusters. Data collected by cluster members (CMs) are transmitted to their cluster heads (CHs) and finally received by the base station (BS). The goal of this network is to maintain an energy neutrality state and maximize the effective data collection of the BS. In reality, a WSN also needs to dynamically meet application-level QoS requirements, that is, to adjust its own operating state according to changes in the deployed environment. The agents share their CH energy information cooperatively within clusters. A series of simulations was conducted to explore the
effect of our approaches on network performance compared to traditional approaches and other RL methods. Simulations show the adaptive power manager can ensure better network performance and avoid node energy depletion.

The main contributions of our work are listed as follows:

1. An adaptive power manager in EH-WSNs by CRL methods is proposed to increase the network throughput by sharing the information between sensor nodes in one cluster, such as residual battery energy and harvesting energy. In cooperative $Q$-learning and SARSA($\lambda$) algorithms, each agent learns from the environment and their CH. It helps the nodes to learn to select the proper transmission rate under the uneven clustering network to maximize the network throughput.

2. The impact of the environmental parameters regarding different initial node energy, transmission power, and the number of sensor nodes in the field is explored.

3. Tuned static policy, stochastic policy, and genetic algorithms (GAs) are compared with the proposed algorithm which successfully provides better network QoSs.

The rest of this article is organized as follows. Section “Related work” presents a brief literature overview on EH-WSNs and RL. Section “System model” gives the whole framework of the system model. Section “CRL model” describes our CRL approaches for the energy manager to adaptively adjust the transmission rate in EH-WSNs. The experimental results and discussion are presented in Section “Simulation and results.” Section “Conclusion” is the conclusion of this article.

**Related work**

This section gives a brief overview of clustered EH-WSN and RL applications in power management of IoT.

**Cluster-based EH-WSNs**

Clustering routing protocols of WSNs have been extensively studied in the literature. They are represented by the low-energy adaptive clustering hierarchy (LEACH)\(^9\) and its following improved algorithms. Their main goals are to achieve energy balance among the whole networks. However, these protocols may lead to the “hot spot” problem in WSNs,\(^10\) that is, the nodes nearing the BS die faster than nodes far from the BS. Uneven clustering algorithm could solve part of this problem. The energy efficient uneven clustering (EEUC) algorithm\(^11\) determines the uneven competition radius according to the distance between the nodes and the BS, so that the size of clusters close to the BS is smaller, saving energy for communication and transmission between clusters. Moreover, in the selection of the relay node, the remaining energy of the node and the distance from the CH to the BS are comprehensively considered. Considering the balancing the whole network energy, Peng and Low\(^12\) proposed a routing protocol based on CH groups, in which CH group members take turns to balance CH energy consumption.

Some works have been done on maximizing network transmission information in WSNs and EH-WSNs, but more investigations under different protocols need to be explored more thoroughly. Wang et al.\(^13\) investigated the mathematical formula to get the optimal transmitting power of nodes and the algorithm for selecting the CH to maximize the data gathering in WSNs. Mehrabi and Kim\(^14\) introduced a mixed-integer linear programming (MILP) optimization model for maximizing network throughput in EH-WSNs. An uneven clustering protocol in solar-powered WSN is proposed in our previous work to maximizing the network throughput with fixed data rate in each node.\(^15\)

**RL applied in power management**

RL algorithms can learn optimal policy by interacting with an environment based on the reward. Value-based approach, such as $Q$-learning and SARSA, and the policy-based approach, such as proximal policy optimization (PPO)\(^16\) using a neural network as a function approximator have been widely used in power management in IoT applications. An adaptive power manager is presented in Shresthamali et al.\(^8\) to adjust duty cycles in solar energy harvesting sensor nodes using the SARSA($\lambda$) method to ensure optimal network performance without draining its battery. It is evaluated at the end of each episode which is believed to achieve the ENO-Max objective better compared to traditional methods. The paper considers the factors on climatic factors and battery degradation. Ortiz et al.\(^17\) proposed methods using RL optimization to achieve energy neutrality for power management. The reward function is based on the distance from the energy neutrality state and the remaining battery of the node. Although there may be a more general formula, this objective is a good rewarding strategy that indicates a certain state or action advantageous, rather than how to achieve the goal. We also use the same metric as part of state evaluation to determine rewards. Aoudia et al.\(^18\) proposed an energy manager to maximize the throughput in a two-hop energy harvesting wireless network. SARSA is
used to solve the problem in the whole network. Hsu et al.\textsuperscript{19} proposed an RL approach named as RLMan as an energy manager according to the current remaining energy level and proved it more robust to the node consumption variation. Hsu et al.\textsuperscript{20} presented an RL-based power manager by a Q-learning algorithm and trained the node to select duty cycles dynamically. Their reward function is energy neutrality state which is the distance from energy neutrality state to energy storage level and the throughput requirements. Rioual et al.\textsuperscript{21} explored different reward functions to find the most appropriate variables in these functions to achieve the proposed performance. Experimental results from comparing the different functions present the effect of reward functions could result in different energy consumption. Given uncertain energy availability, Fraternali et al.\textsuperscript{22} used RL algorithms to maximize the sample rate with available harvested energy. Most of the previous works use RL in regulating one single node duty cycles dynamically in IoT, without considering interactions between nodes affecting the network performance.

Multi-agent reinforcement learning (MARL) is a system where several agents interact in the same environment. The multi-agent environment represents as \(<N, S, A, R, P, O, R>\), where \(N\) is the agent set, \(S\) is the state space, \(A = \{A_1, \ldots, A_N\}\) is a set of actions of the agents, \(P\) is the state transition probability, \(R\) is the reward function, and \(O = \{O_1, \ldots, O_N\}\) is the observation of all agents. In any type of environment, \(A\) represents the action vector of one agent, \(a_{-i}\) is the vector of all agents except agent \(i\), and \(\tau_i\) represents the action history of agent \(i\). Besides, \(T, S, A\) and \(P\) are the environment transition matrix, the state space, and the action space, respectively.

Each agent makes certain actions at each time step and works together with other agents to achieve the goals. Due to the complexity of the MARL environment and the characteristics of multi-agent collaboration, the complexity of training multi-agents is relatively high, and most MARL problems are classified as NP-Hard problems.\textsuperscript{23} which are difficult to find the solution algorithm of polynomial time complexity. The factors that need to be considered when modeling MARL systems include (1) centralized or decentralized control, (2) a fully or partially observable environment, and (3) a cooperative or competitive environment. In the centralized control mode, the centralized controller is responsible for each agent action in each time step. However, in a decentralized system, each agent’s behavior is determined by itself. Moreover, agents can cooperate to achieve common goals or may compete with each other to maximize their rewards. Under different circumstances, an agent may access all the information of other agents, or each agent may only observe its local information.

In the problem of cooperative environment, if an agent can fully observe the environment, each agent \(i\) observes the global state \(s_i\) at time step \(t\) and uses the local random strategy \(\pi_i\) to take the action \(a_i^t\) and obtain the reward \(r_i^t\). If the environment is fully cooperative, then every agent has a common reward value \(r_i\) at every time step, that is, \(r_i^t = r_1^t \cdots = r_n^t = r\). If the agent cannot observe the system state completely, each agent only accesses its local observation value \(o_i^t\). The modification of the Bellman equation corresponding to the MARL environment is shown in equation (3)\textsuperscript{24}.

\[
Q_i^t(s, a_i | \pi_{-i}) = \sum_{a_{-i}} \pi_{-i}(a_{-i}, s) \left[ r(s, a_i, a_{-i}) + \gamma \sum_{s'} P(s' | s, a_{-i}) \max_{a_i'} Q_i^{t+1}(s', a_i') \right]
\]

where \(\pi_{-i} = \prod_{j \neq i} \pi_j(a_j | s)\) represents the strategy of all agents except the agent \(i\).

Although the MARL method has been successful in practical applications, MARL has the nature of multi-agent combination problem, that is, its complexity will increase exponentially with the number of agents \(N\) (\(N\) may be thousands or even larger). Independent Q-learning (IQL)\textsuperscript{25} considered the most intuitive collaborative RL method avoids the problem of poor scalability where each agent treats other agents as part of the environment and independently learns its strategy. But the environment of IQL becomes unstable since it contains other learning agents; therefore, convergence cannot be guaranteed. However, a lot of empirical evidence shows that IQL usually achieves good results in practice.\textsuperscript{26} Littman\textsuperscript{28} studied the convergence characteristics of multi-agent reinforcement learning\textsuperscript{27} and believed that in a competitive environment, players can act the best performance by \(Q\)-learning. In a collaborative environment (such as in a cooperative game, all agents share the same reward function), it is necessary to make strong assumptions about other agents to ensure convergence\textsuperscript{28} such as Nash \(Q\)-learning\textsuperscript{29} and Friend-or-Foe \(Q\)-learning.\textsuperscript{30} There is no known research on the guaranteed convergence of value-based RL algorithms in other types of environments.\textsuperscript{28}

There are several works in which the authors used CRL methods to optimize the performance of power management. Khan and Rinner\textsuperscript{31} proposed a CRL method for determining and scheduling tasks based on its behavior and neighboring nodes’ information. Therefore, application performance and energy achieve a better trade-off. Emre et al.\textsuperscript{32} introduced the so-called cooperative \(Q\) method which makes radios learn and adapt to their environment by sharing their information. Its proposal aimed to maximize energy efficiency while ensuring buffer occupancy below the predetermined level. Wu et al.\textsuperscript{33} proposed a cooperative \(Q\)-learning method to investigate the energy management
approach to optimize the network throughput. The network works on a preset routing protocol where the relay node receives data packets from the previous node and then transmits it to the next one. The whole process repeats until the data are finally sent to the BS successfully. A node regulates its duty cycle according to its neighbor’s energy information. Simulations show that their proposed scheme can work effectively to satisfy the network performance requirements. However, no previous research works have investigated the algorithms in a clustered WSNs where energy has already been balanced and the optimization of the duty cycle has more challenges in this environment.

System model

Figure 1 shows an overview of our system model and energy management approach. The adaptive energy manager uses CRL to monitor and regulate their duty cycles to keep the nodes in an energy neutral state. This is a typical clustered EH-WSN which has a BS and sensor nodes randomly located in the field. Each sensor node consists of energy harvesting equipment (solar panel) to complement energy, an energy buffer (supercapacitor), and a power manager for each node (agent). They are divided into clusters periodically. The CHs of clusters are rotated throughout the system life cycle, for preventing them from replenish their limited energy. Once the CHs lose all the energy, they are no longer operational and all the CMs belonging to the cluster lose their communication capabilities. All the CHs collect data sent by the CMs within their clusters, then aggregate and forward these data to the BS. We assume that the BS can access an unlimited amount of energy. The agent can adjust its duty cycle according to the node and environmental state. To achieve the maximum overall throughput during a period of operation time (usually a whole day), we also utilize the periodical predicted harvesting energy information of each node from the energy predictor proposed in Ge et al.\(^3\)

The uneven clustering protocol is deployed in our system. For each node \(S_i\), the location \((x_i, y_i)\) is known. The field area is \((0,0)-(L,L)\). The coordinates of the BS are \((L/2, L/2)\) and the distance \(d(S_i, BS)\) from each node to the BS can be calculated. The schemes of cluster formation and CHs reselection strategies are described in the following three steps:

Step 1. Tentative CH selection: at the beginning of each round, nodes are randomly selected as tentative CHs, if \(\text{rand}() \leq p/(1 - p \times \text{mod}(r, \text{round}(1/p)))\), where \(p\) indicates the probability of being CH and \(r\) indicates the number of rounds.

Step 2. CH competition: for any tentative \(CH_j\), it determines the competition area \(R_c\) in equation (4)

\[
R_c = R_0 \times \left(1 - c \times \frac{d_{\text{max}} - d(S_i, BS)}{d_{\text{max}}}\right)
\]

where \(R_0\) is the maximum competition radius in the network, \(d(S_i, BS)\) is the distance between the node \(S_i\) and the BS, \(d_{\text{max}}\) is the maximum distance of the node to the BS, and \(c\) is a constant coefficient (between 0 and 1). The definition of \(R_c\) guarantees that clusters closer to the BS have less competition range. In this competition range, only one CH with the highest residential energy and predicted harvesting energy in the next time slot becomes the final CH.

Step 3. Cluster formation: each node calculates the distance to all the CHs and joins the closest CH.

The energy consumption model from LEACH\(^7\) is adopted in the network, including energy consumed during transmission, reception of packets, and data dissemination. The details are described in section “Simulation and results.”

CRL model

Framework of CRL

Our framework of CRL defines a group of cooperative sensor nodes under the EH-WSN environment where action and state space are modeled. Agents are trained in each round of cluster formation. Each node harvests a certain amount of energy in one time slot. The current state of each node consists of the residual energy, the solar energy that could be harvested in the next time slot, and its CH residual energy. At the end of each round, the agent will receive some rewards for evaluating its behavior and upgrading its strategy (\(Q\)-table). The agent keeps taking a series of actions and traverses the alternating state until the end of the episode. The state definition, the action space, and the reward scheme are described below.

Environment. In the EH-WSN, the whole network tries to achieve energy balance and high network throughput. Sensor nodes can supplement energy from their solar panel. In each round, clusters are formed periodically. Each CM transmits the data to its CH, and then the CH will aggregate the data from its CMs and then send all the data to the BS. For energy efficiency, the energy manager would prevent the node from depletion completely. Each node will determine the transmission
rate to be set based on the predicted energy and the remaining energy for the next time slot.

**Agent.** The agent of each node communicates with the environment and takes the power management action. The agent is trained by a series of episodes. Each episode has 24 hours and 10 times of cluster reformation per hour, therefore there are 240 rounds per episode. The node transmission process of each round includes the information exchanging stage and data transmission stage. In each round of information exchange stage, each agent in the cluster can receive CH information, including energy information.

**Action.** The action space $A$ is defined as the choice of the transmission rate of sensor nodes per round. The selected action corresponds to the packages that the node will try to send during one round. Discretization of the action space reduces the computation time and therefore makes the CRL algorithm achieve faster convergence. Action space $A$ is discretized where $A \in (R_{\text{min}}, R_{\text{max}})$ with $R_{\text{min}}$ and $R_{\text{max}}$ being the minimum and maximum data transmission rates of the sensor nodes, respectively. In our case, the action space is defined as 12 different transmission rates, which are $A = \{4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26\}$. Under the clustered EH-WSN, one timeslot has 60 min and there are 10 cluster re formations per time slot. After the reselection of CH, the transmission rate is reset for each node according to its $Q$-table.

**State.** In order to achieve long-term stability of the network and high transmission of information, the nodes in the energy harvesting WSN need to determine the data transmission rate based on their own remaining energy and the energy that can be harvested in the next cycle, and it is related to the energy level of the next relay node. The current state of the energy level of the non-CH node becomes an important factor to determine the data transmission rate. The observation state of each node consists of three observations in equation (5)

$$S_t = [E_{\text{harvest}}(t_k), E_{\text{dis}}(t_k), E_{\text{cdis}}(t_k)]$$  \hspace{1cm} (5)

$E_{\text{harvest}}(t_k)$ shown in Table 1 represents the predicted harvested energy of the node for the next round at time $t_k$. $E_{\text{dis}}(t_k)$ shown in Table 2 represents the state of the nodes’ distance from the currently remaining energy to the initial energy $B_0$. $E_{\text{cdis}}(t_k)$ shown in Table 2 represents the state of its CH’s distance from the remaining energy to its initial energy. If the energy level of a node at time $t_k$ is $B(t_k)$, the distance from the node to its initial energy $E_{\text{dis}}(t_k)$ is calculated in equation (6). Similarly, the residual energy of its CH which receives the packets that are sent by the node is calculated according to equation (7)

$$E_{\text{dis}}(t_k) = B(t_k) - B_0$$  \hspace{1cm} (6)

$$E_{\text{dis}}(t_k) = B(t_k) - B_0$$  \hspace{1cm} (7)

where $B(t_k)$ represents the energy remaining in the current node, and $B_0$ is the ideal energy buffer (set as the initial buffer in our case).
Table 1. State of $S_{\text{harvest}}(t)$.

| $S_{\text{harvest}}(t_i)$ | Harvested energy (J) |
|---------------------------|---------------------|
| 0                         | $E_{\text{harvest}}(t_i)<0.0001$ |
| 1                         | $E_{\text{harvest}}(t_i)<0.1$ |
| 2                         | $E_{\text{harvest}}(t_i)<0.22$ |
| 3                         | $E_{\text{harvest}}(t_i)<0.28$ |
| 4                         | $E_{\text{harvest}}(t_i) \geq 0.28$ |

Table 2. State of $S_{\text{dis}}(t)$ and $S_{\text{edis}}(t)$.

| $S_{\text{dis}}(t_i), S_{\text{edis}}(t_i)$ | Distance from remaining battery energy to initial energy (J) |
|-------------------------------------------|----------------------------------------------------------|
| 0                                         | $E_{\text{dis}}(t_i)<0.01$ |
| 1                                         | $E_{\text{dis}}(t_i)<0.5$ |
| 2                                         | $E_{\text{dis}}(t_i)<1.0$ |
| 3                                         | $E_{\text{dis}}(t_i)<2.2$ |
| 4                                         | $E_{\text{dis}}(t_i) \geq 2.2$ |

$$E_{\text{edis}}(t_k) = B_c(t_k) - B_{c0}$$

where $B_c(t_k)$ represents the energy remaining in the CH node, and $B_{c0}$ is the initial energy buffer.

**Reward.** The objective function of our system is to maximize the throughput and balance the energy. The goal of the system is to send as much data as possible under limited energy resources while keeping the node in an energy neutral state and avoiding less number of energy depletion of nodes. The reward calculated in equation (8) is then designed for achieving this goal:

$$\text{Reward} = \begin{cases} -100 & \text{if energy level reaches 0 (node dies)} \\ -1 & \text{if a packet fails to be sent to the BS} \\ 1 & \text{if a packet is sent to the BS successfully} \end{cases}$$

**Cooperative Q-learning algorithm**

$Q$-learning is a classic value-based off-policy RL algorithm. It is expected that taking action $a (a \in A)$ under a state $s (s \in S)$ at a certain moment can obtain reward $Q(s,a)$, the environment responds to the corresponding reward $r$ according to the action of the agent. The algorithm constructs a $Q$-table of state–action pairs to store $Q(s,a)$, and selects the action that can obtain the greatest reward according to the $Q$-value. When applying the $Q$-learning algorithm under MARL, as an iterative algorithm, the time complexity will explode with the scale of the problem. One of the direct and effective algorithms is IQL [25], in which each agent is trained independently. To avoid the scalability problem, a cooperative $Q$-learning algorithm shown in Algorithm 1 is designed and applied to adaptively adjust the transmission rate of nodes in the EH-WSN environment.

Algorithm 1 illustrates the steps of the cooperative $Q$-learning algorithm. After initializing the $q_{\text{table}}$ of each node to a zero matrix (line 1), the node state (i.e. residual energy, CH residual energy, and predicted energy) is discretized and received by the agent (line 2). The algorithm then goes into a while loop. Each CH sends its parameters to its CMs (line 4). According to its state (line 5), each node starts to select an action (line 4) which follows the $\varepsilon$-greedy policy (line 6–9). Then, each node in the cluster changes its transmission rate and the reward is returned accordingly (line 10). The next state of each node is observed in each round (line 11). The $Q$-table of each agent is updated separately according to the state, the next state, and selected action (line 12–14). $q_{\text{predict}}$ is the $q$-value of taking action $a$ in the state $s$. $q_{\text{target}}$ is the maximum value from taking the actions in the $q$-table multiplied by the discount factor $\gamma$ with the addition of the reward $r$ when reaching the state $s'$. The difference between $q_{\text{predict}}$ and $q_{\text{target}}$ multiplied by learning factor $\alpha$ is then added to the corresponding $q$-value in each node’s $q$-table.

**Computational complexity analysis.** Considering the computational complexity of each iteration in the MARL environment, since each node maintains its own $q$-table, the highest computing burden is attributed to the state–action search of the $q$-table and the update of the corresponding state. Algorithm 1 uses the $\varepsilon$-greedy strategy for each agent node to select actions in a certain joint state. The time complexity for searching the node’s $q$-table and updating $q$-table is $O(|A||E_{\text{harvest}}||E_{\text{edis}}|^2)$, where $|A|$ is the number of optional actions, $|E_{\text{harvest}}|$ is the number of discrete states of energy that can be collected, and $|E_{\text{edis}}|$ is the number of states of distance from the energy neutral state. The time complexity for search will exceed the tolerable range when too large or
infinite action or state spaces exist, and additional methods such as function approximation are required. In the current environment, due to the discretization of the node energy space and the limited state of the action space, the time complexity is relatively low. The computational complexity of the algorithm is polynomial to $|A|$, $|E_{harvest}|$, and $|E_{dis}|$ with good scalability.

**Convergence analysis.** The $Q$-learning algorithm was first proposed in the literature.\(^3\) It is a major breakthrough in RL because this is the first RL algorithm that guarantees to converge to the optimal strategy. Its convergence is proved in the literature.\(^3\) In a certain Markov decision process (MDP), the $Q$-learning algorithm can be equivalently defined as

$$Q_0(s, a) = 0 \text{ for all } s \in S, a \in A$$

$$Q_{t+1}(s, a) = \begin{cases} Q_t(s, a), & \text{if } s \neq s_i \text{ or } a \neq a_i \\ r(s, a) + \gamma \max_{a' \in A} Q_t(b(s, a), a'), & \text{otherwise} \end{cases}$$

(9)

The individual $Q$-learning algorithm is known to converge to the optimal $Q'(s, a)$ that maximizes the reward.\(^3\) $Q$-learning finds a better strategy $\pi$ through each iteration until it reaches the best strategy $\pi'$. $Q$-learning can guarantee convergence when two conditions are met, that is, after an arbitrarily long time, the strategy will be arbitrarily close to the optimal strategy (note that the following conditions do not guarantee how fast the strategy will approach the optimal strategy):

1. It is required that the sum of learning rates must diverge, but the sum of squares must converge as shown in equation (10)

$$\sum_{k=1}^{\infty} \alpha_k = \infty \text{ and } \sum_{k=1}^{\infty} \alpha_k^2 < \infty$$

(10)

2. Each state-action must be accessed unlimited times: each action in each state must be selected by the strategy with non-zero probability, that is, $\pi(s, a) > 0$ for all state-action pairs. In fact, using the $E$-greedy strategy (where $\epsilon > 0$) can ensure that this condition is met.

Through the previous analysis, the value function $Q_*(S, A_*)$ of the known centralized control method (similar to a single agent environment) increases monotonically with time $t$ and can converge to the optimal solution $Q'$. Although in a multi-agent environment, because the model depends on the joint policies of other learning agents, it has not been proven to converge in a general sense. However, in the current cluster routing environment, as a specific MARL environment, each CH and its CMs can communicate. Therefore, the CH node status can be observed by its CMs, but the actions of CMs cannot be observed by other CMs. After each agent’s actions are executed in the environment, the rewards received by each agent are returned. First, prove that the local action-value function $q_i(S, a_i)$ of a single agent is a mapping of the value function $Q_i(S, A_i)$ of the centralized control mode, which can select the highest value function for the intelligent individual. Similar to the distributed $Q$-learning based on IQL extension,\(^7\) it is a decentralized and fully cooperative multi-agent problem that can completely observe the system state. The algorithm updates the $Q$-value when the $Q$-value is guaranteed to be improved, that is, for a state $s$, and a joint action $a = (a_1^i, \ldots, a_n^i)$, for the agent $i$, there are

$$q_{t+1}(s, a) = \begin{cases} q_t(s, a), & \text{if } s \neq s_i \text{ or } a \neq a_i \\ \max \{ q_t(s, a), r(s, a_1^i) + \gamma \max_{a_1^i \in A} q_t(s_1^i, a_1^i) \}, & \text{otherwise} \end{cases}$$

(11)

**Proposition 1.** For the deterministic MDP model $M = (S, A, P, R)$, with $n$ agents, the collaborative $Q$-learning method is adopted, where each agent has the same reward function $r_1 = \cdots = r_n = r, r \geq 0$. Then, when the agent $i$ adopts the individual action $a_i^t$ at time $t$, then equation (12) holds

$$q_{t+1}(s, a_i^t) = \max_{a_i = (a_1^i, \ldots, a_n^i)} Q_i(s, a_i)$$

(12)

where $Q_i(s, a_i)$ and $q_{t}(s, a_i^t)$ are the action-value function of the centralized control mode and the local action-value function of the agent $i$, respectively; $Q_i(s, a_i)$ and $q_{t+1}(s, a_i^t)$ are defined in equations (9) and (11), respectively.

**Proof**

$t = 0$: values in the tables are initialized as 0, then equation (12) holds;

$t \mapsto t + 1$: at any time $t$, all agents take action $a_i = (a_1^i, \ldots, a_n^i)$. For agent $i$, equation (11) holds, then:

1. When $s \neq s_i$ or $a \neq a_i^t$, equation (12) holds because equation (9) or equation (11) is not updated.

2. When $s = s_i$ and $a = a_i^t$, since equation (11) holds, then
\( q'_{t+1}(s, a') = \max \{ q'_t(s, a'), r(s, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} q'_t(s_{t+1}, a_{t+1}) \} \)

Assume that equation (12) holds at time step \( t \), then

\[ q'_{t+1}(s, a'_t) = \max \left\{ \max_{a_t} Q_t(s_t, a_t), \max_{a_{t+1}} \left( r(s_t, a_t) + \gamma \max_{a_{t+1}} Q_t(s_{t+1}, a_{t+1}) \right) \right\} \]

Because the \( Q \)-table is monotonically increasing

\[ q'_{t+1}(s, a'_t) = \max_{a_t} Q_t(s_t, a_t), Q_t + 1(s_t, a_t) \]

Therefore, equation (12) holds at \( t + 1 \), that is, multi-agent \( Q \)-learning in this situation is consistent with the centralized control \( Q \)-learning method, and it can also achieve convergence in this environment.

**Cooperative SARSA(\( \lambda \)) algorithm**

SARSA algorithm is an on-policy algorithm which is a temporal difference learning approach to solve RL control problems. Given five elements of an RL environment: state space \( S \), action space \( A \), instant reward \( R \), attenuation factor \( \gamma \), exploration rate \( E \), solving the optimal action-value function \( q^* \) and the optimal strategy \( \pi^* \). Based on the \( \varepsilon \)-greedy method, an action \( a_t \) is selected at the current state \( s_t \) at time \( t \), and the system turns to a new state \( s_{t+1} \), and at the same time receives an instant reward \( r_t \), in the new state \( s_{t+1} \). The \( \varepsilon \)-greedy method selects an action \( a_{t+1} \) in the state \( s_{t+1} \). This action \( a_{t+1} \) is not executed at this time, and is only used to update the value function. The value function update formula

\[ Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha (r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)) \]

where \( \gamma \) is the discount factor and \( \alpha \) is the learning factor.

SARSA(\( \lambda \)) algorithm is the alternation of standard SARSA to improve the algorithm. After each action is taken in SARSA, only the previous step is updated which is SARSA (0) while SARSA(\( \lambda \)) can update the step before the reward by having the trace decay factor \( \lambda \in [0, 1] \). The same as the value of the expected \( Q \) in the future, \( \lambda \) is the opportunity for the agent to look back at the previous path when updating the \( Q \)-table after the agent receives the reward. Equivalently, the agent inserts a flag on the ground every time the agent takes a step, and then the flag becomes smaller every time the agent takes a step. Define \( \lambda \)-return as the sum of the gains of all steps \( n \) from 1 to \( \infty \) times the weight. The weight of each step is \((1 - \lambda)\lambda^{n-1}\), so the calculation formula of \( \lambda \)-return is expressed as

\[ G^\lambda_t = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G^{(n)}_t \]  

Comparing with SARSA, it has an additional matrix \( E(s,a) \) (eligibility traces) in the algorithm which is used to save the contribution of each step in the path to the subsequent reward which is defined as

\[ E_0(s,a) = 0 \]

\[ E_t(s,a) = \begin{cases} \gamma \lambda E_{t-1}(s,a) & \text{if } (s,a) \neq (s_k,a_k) \\ \gamma \lambda E_{t-1}(s,a) + 1 & \text{if } (s,a) = (s_k,a_k) \end{cases} \]  

SARSA(\( \lambda \)) can learn the optimal strategy more quickly and effectively. Because the action \( a \) in the \( Q \)-table may be mapped to several different states, some \( Q \)-values may indicate that action \( a \) is good, and some may indicate that \( a \) is not good. The first few steps are updated only after the reward is definite for more clearly evaluating the quality of the action. Therefore, under the appropriate \( \lambda \), SARSA(\( \lambda \)) converges faster.

We apply cooperative SARSA(\( \lambda \)) for adaptive power management in EH-WSNs. Each node has a common goal, which is to maximize the network throughput. In time period \( t \), each node makes a decision individually. Each node maintains its own \( q^\lambda \) table and updates the \( q^\lambda \) table according to equation (16). Algorithm 2 shows the pseudo code of the cooperative SARSA(\( \lambda \)) algorithm

\[ q'_{t+1}(s, a) = \begin{cases} q'_t(s, a), & \text{if } s \neq s_t \text{ or } a \neq a'_t \\ \max(q'_t(s_t, a_t), (1 - \alpha)q'_t(s_t, a_t) + \alpha (r(s_t, a_t) + \gamma q'_t(s_{t+1}, a_{t+1}))), & \text{otherwise} \end{cases} \]

Algorithm 2 describes the steps of the cooperative SARSA(\( \lambda \)) algorithm. After initializing the \( q \)-table of each node to a zero matrix (line 1), the state of each node (node residual energy, its CH residual energy, and predicted energy information) is discretized (line 2). After clustering formation, the CH information is sent to each CM (line 4). Each node in the cluster selects its own action according to its \( q \)-table and calculates the reward accordingly (line 8). In each round, the next
state of each node is observed (line 9). According to the current state, the next state, and the selection action, the \( q \)-table and the eligibility trace matrix will be updated individually (lines 10–15).

Similar to the collaborative \( Q \)-learning method, the collaborative SARSA algorithm considers the computational complexity of each iteration. The complexity of energy management policy based on clustered EH-WSNs is independent of the number of nodes \( N \). Algorithm 2 uses the \( \epsilon \)-greedy strategy for each agent node to select actions in a certain joint state. The time complexity for searching the node’s \( q \)-table and updating \( q \)-table is \( O(|A||E_{\text{harvest}}||E_{\text{dis}}|^2) \), where \( |A| \) is the number of optimal actions, \( |E_{\text{harvest}}| \) is the number of discrete states of energy that can be collected, and \( |E_{\text{dis}}| \) is the number of states of distance from the energy neutral state. In the current environment, the computational complexity of the algorithm is polynomial to \( |A| \), \( |E_{\text{harvest}}| \), and \( |E_{\text{dis}}| \).

In SARSA, when searching for the best solution, the strategy \( \pi \) is updated. Evidence shows that select the action \( \pi'(s) = \arg \max_a q_\pi(s,a) \) to update can always improve the strategy or achieve the best strategy. The proof of the convergence of the SARSA method can be found in the literature.\(^{36}\) The convergence analysis of the cooperative SARSA algorithm is similar to the \( Q \)-learning method in section “Cooperative \( Q \)-learning algorithm.”

**Simulation and results**

A clustered EH-WSN network environment is carried out on Python. The simulation environment constitutes the system environment of sensor nodes that continuously communicate with the adaptive energy manager implemented in the CRL algorithm. Agents select actions (data transmission rate) to adjust the nodes’ state in order to optimize network data transmission. In the following sections, energy data resources, the model configuration of the network, and case studies are described in detail.

**Energy source**

The data to simulate the harvested solar energy are retrieved from the US Department of Energy, National Solar Radiation Database\(^{39}\) which contains historical solar data with a 1-h sample rate and statistics results in more than 1000 locations of the United States. Data from different locations, including Michigan, Alabama, and Nevada, are used in the experiments in order to compare the performance of RL methods. Figure 2 shows 7 continuous days of global horizontal solar radiation profile in a site of Alabama in 2008. It shows different amounts of hourly energy available for harvesting under the different weather conditions where January 17 and 18 have obvious lower harvesting resources available than the other 5 days.

**EH-WSN clustering and energy-related parameters**

In the sensor network field area, the example of a clustered solar power network with 60 sensor nodes is depicted in Figure 3. The field is a square with the length of 200 m. Each node uses a solar panel device to complement energy. The stationary BS with the “X” shape marker in our system model is deployed in the middle of the WSN field by which a better performance can be achieved.\(^{40}\) The BS could get access to an unlimited amount of energy and have strong computational power. All the nodes are grouped periodically into different clusters by the same color. Each cluster is composed of a CH indicated by a triangle marker and some CMs with dot markers. Data are sent by the CMs to their CHs, then aggregated and forwarded to the BS. Section “System model” describes the steps of the uneven cluster formation algorithm in detail. Figure 3(a) and (b) forms five different clusters, respectively, in different rounds. These two graphs show different clustering methods based on the same distribution of nodes, including the number of nodes and position coordinates. The positions of the CHs have changed as the CH rotation of the network during the initialization phase of each round, which is conducive to balancing energy load and prolonging the network life.

The energy consumed in the simulation system is calculated according to the energy consumption model in Heinzelman et al.\(^9\) Assuming that the model is ideal, energy consumption—such as data acquisition and data storage—is constant. Besides it is assumed that there is no static energy consumption which can be set to 0 in the simulation, and the energy consumed by the battery itself is ignored. For simplicity, assume it is an ideal battery. The initial battery size of the sensor node is initialized as 0.5 J. For the other parameters in the system: the wireless sensor deployment area is a 200 m \( \times \) 200 m rectangle and the BS is located at the middle of the area with the coordinates (100 m, 100 m). The total number of sensors is 20, 40, and 60, respectively, in different cases. The probability of the sensor being selected as the CH is 0.2. The size of the control packet for communication between the nodes is 500 bits/packet, the size of the data packet is 4000 bits/packet, \( e_s = 1.0 \times 10^{-11} \), \( e_{mp} = 1.3 \times 10^{-15} \), and \( E_{DM} = 5 \times 10^{-09} \). For the threshold \( d_0 = \sqrt{e_s/e_{mp}} \). Communication-related parameters in our approach are correspondent to similar works in the literature summarized in Table 3. Under these settings, simulations are carried out on the Python platform.
**Algorithm 2.** Cooperative SARSA(λ).

1: initialize $q_{table}$ for each node and $E[s,a] = 0$, for all $s,a$
2: set the initial state $s = \text{discretized} \quad ([E_{harvest}, E_{dis}, E_{cdis}] = [0, 0, 0])$
3: for each episode do
4: after the cluster formation, CH parameters ($E_{cdis}$) are sent to CMs
5: for each agent $i$
6: $s_i = \text{discretized} \quad ([E_{harvest}, E_{dis}, E_{cdis}])$
7: $a_i = \text{argmax} \quad q_{table}[s_i,:]
8: get reward $r$ from the EH-WSN environment
9: observe the new state $s'$ and choose the action $a'$ from $Q$-table with $e$-greedy policy
10: $\delta = R + \gamma * q_{table}[s',a']$ $- q_{table}[s,a]$
11: $E[s,a] += 1$
12: for all $s$ belong to $S$, $a$ belong to $A(s)$
13: $q_{table}[s,a] += \alpha * \delta * E[s,a]$
14: $E[s,a] = \gamma * \lambda * E[s,a]$
15: $s = s'; a = a'$

---

**Tuning parameters**

The system has several parameters to be set including clustering parameters (parameters to control the cluster size $R_0$ and $c$ in equation (4)) and CRL parameters (the learning rate $\alpha$, the discount factor $\gamma$, and the trace decay factor $\lambda$ in SARSA). The competition radius of a CH varies according to equation (1) where parameters are tuned empirically through simulation. $c$ between 0.2 and 0.3 leads to better network throughput which is set as 0.2 in later experiments, and $R_0$ is tuned to be a constant 90 m under the field of 200 m $\times$ 200 m.\textsuperscript{15} According to Karray et al.,\textsuperscript{41} the other parameters are relevant to the energy harvesting platform.

The learning rate $\alpha$ determines how fast the model moves to the optimal value and is a trade-off between the previous and current learning results. Too low value of $\alpha$ may cause agents to only care about the previous knowledge and cannot accumulate new rewards. Too high value of $\alpha$ may cause excessive fluctuations of $Q$-values during training. Numerous studies start commonly with a high learning rate so that learning results can change quickly, and then over time, the learning rate is reduced. Under our environment, Figure 4 shows when $\alpha$ is 0.01, there is no improvement within 200 iterations; when $\alpha$ is 0.05, the higher performance of the network is achieved; when $\alpha$ is 0.1, the system also converges faster; when $\alpha$ is set to gradually decrease from 0.1 to 0.05, the experimental performance is similar to when $\alpha$ is set to 0.1.

The discount factor $\gamma$ accesses the impact of future reward value compared to immediate rewards which is usually set between 0 and 1. When the discount factor $\gamma$ is 0, only the current reward is considered, adopting a short-sighted strategy. The larger the value of $\gamma$, the more future benefits we have considered the value generated by the current behavior. If the value of $\gamma$ is too large, the consideration is too long-sighted, even beyond the range that the current behavior can influence which will affect the convergence of the algorithm. Empirical experiments show how the discount factor impacts the training performance of the whole network throughput. When $\gamma$ is higher than 0.9, the performance of the system varies rapidly during the process. When $\gamma$ is less than 0.7, the network throughput is comparatively low and difficult to achieve.

---

**Figure 2.** Solar radiation profile in Alabama 2008.

**Figure 3.** Examples of EH-WSNs: (a) one formation of five clusters and (b) another formation of five clusters.
comparatively high network QoS. Considering these results, $\gamma$ is set as 0.9.

When the agent chooses actions based on its own $q$-table, it uses the $\varepsilon$-greedy algorithm. If $\varepsilon = 0$, a completely greedy method means the highest $q$-value is always selected among all $q$-values in a specific state, which may lead to a local optimal state. The randomness of $\varepsilon$ is necessary for exploration. In the experiment, $\varepsilon$ is set to 0.1 which means the agent chooses a random action with a probability of 0.1.

The trace decay parameter $\lambda$ in eligibility traces in SARS$\lambda$($\lambda$) controls how eligibility decays over time. Through experiments, when $\lambda$ increases, SARS$\lambda$($\lambda$) with a higher $\lambda$ converges quickly at the first stage of training to than with a lower $\lambda$. But when $\lambda$ is 0.9, some lower performance is observed. In order to balance the trade-off between good performance and convergence, the trace decay $\lambda$ is regulated as 0.8.

### Results and discussions

The performance of the EH-WSN network is measured by the number of data packets successfully transmitted to the BS of the network in one whole day. Distributions of nodes of the networks are generated randomly in each simulation.

**Performance evaluation.** Figure 5(a) and (b) shows the network throughput of $Q$-learning and SARS$\lambda$($\lambda$) algorithm results, respectively, in its optimal policy when all 60 nodes in the field are trained over 700 iterations. It is obvious that the total number of data packets transmitted by the network increases as the network is trained by the CRL, and then the data oscillate around 170,000 packets to 218,000 packets. This shows that in the current network configuration, the optimal performance of the strategy is 218,000 packets. Cooperative $Q$-learning and SARS$\lambda$($\lambda$) achieve similar performance. Figure 6 shows the node transmission rate changes of a node in one iteration both at the beginning of $Q$-learning training and after 500 iterations. The supplement energy charges the nodes every 4 h. Figure 6(b) and (d) shows the battery residual energy at the beginning of the training and 500th iteration, respectively. At the beginning of the training shown in Figure 6(a), the transmission rate is quite random and after 500 iterations’ training, the

### System parameter settings

| Parameters                             | Value                        |
|----------------------------------------|------------------------------|
| Data packet size                       | 4000 bits/packet             |
| Control packet size                    | 500 bits/packet              |
| Energy for modulation, filtering, and spreading data ($e_{md}$) | $5 \times 10^{-8}$ J/bit   |
| Energy for receiving one bit of data ($e_{r}$) | $5 \times 10^{-9}$ J/bit   |
| Energy for amplifier ($e_{mp}(d < d_{0})$)                | $1.0 \times 10^{-11}$ J/(bit$\cdot$m$^2$) |
| Energy for amplifier ($e_{mp}(d \geq d_{0})$)                | $1.3 \times 10^{-15}$ J/(bit$\cdot$m$^4$) |
| Energy for data aggregation ($E_{DA}$)                     | $5 \times 10^{-9}$ J/bit   |

**Table 3.** System parameter settings.

---

![Figure 4. Different learning rate comparison.](image1)

**Figure 4.** Different learning rate comparison.

![Figure 5. CRL iterations: (a) cooperative $Q$-learning and (b) cooperative SARS$\lambda$($\lambda$).](image2)

**Figure 5.** CRL iterations: (a) cooperative $Q$-learning and (b) cooperative SARS$\lambda$($\lambda$).
node is trained to set at 22 packets/round since the harvested energy is abundant as shown in Figure 6(c). The nodes could operate continually by the stored energy at night time (around round# 180 to 240).

Comparison with static policy. Under the static transmission rate, the network data transmission performance is shown in Figure 7 which indicates that the performance of the network is sensitive to the predefined data transmission rate, that is, the network performance at different data transmission rates varies greatly. Figure 7(a)–(d) shows under static transmission rate when the transmission rate is set as 8, 12, 16, and 20 packets per round. In Figure 7(d), the network performance of this certain case has a high deviation under different clustering formation and has little chance to have good network performance since the energy cannot maintain the transmission rate as high as 20 packets per round. According to Figure 8(a), under static data transmission, 18 packets/round for each node are optimal in the current configuration. When the transmission rate is less than 18 packets/round, the node energy is relatively abundant, and the performance of the network increases as the rate increases. When the transmission rate is greater than 18 packets/round, the high data transmission rate makes the node energy consumption too quick, causing some nodes to be exhausted and become dead. Thus, the network performance declines dramatically under 20 packets/round. Figure 8(b) shows the same trend that the number of dead nodes increases rapidly especially when the transmission rate is above 18 packets/round. The performance of the network depends on the specific network configuration (including system parameters) with the optimal transmission rate. However, it is not always able to know the optimal transmission rate for each network configuration, and tracking changes in network parameters and environments is particularly difficult in the case of constantly changing network parameters and environments which limits the performance of the whole network. The cooperative Q-learning strategy is obviously better than the best performance of the optimal static strategy with 213,000 packets. Considering the adaptability of the strategy to the environment and the optimal strategy, using manually configuration is not applicable, so the network may not be able to run at the optimal rate. To compare different approaches under different network environments, more experiments have been done on cooperative Q-learning and SARSA(λ).

Comparison with stochastic policy and GA. To evaluate the performance of the proposed algorithm, a stochastic policy was also implemented. A stochastic policy simply enumerates every possible combination in the
solution space, evaluating each and keeping the record for the best solution to date. Table 4 shows one possible solution for one round of energy management in EH-WSNs. However, the efficiency of stochastic policy is extremely low. The solution is represented as a two-dimensional array with height as node dimension and width as time slot dimension. Each cell in the array is assigned one value from the set of \{4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26\}. If there are 60 nodes in the field, a total number of $12^{60 \times 240}$ different combinations. Within 500 iterations and 10 different distributions for 20, 40, and 60 nodes in the field, the performance of random search and collaborative Q-learning is shown in Table 5. The results show that in one case of 40

Figure 7. Network throughput by static transmission rate: (a) packet rate = 8, (b) packet rate = 12, (c) packet rate = 16, and (d) packet rate = 20.

Figure 8. Network performance under different transmission rates: (a) network throughput under different constant transmission rates and (b) number of dead nodes under different constant transmission rates.
nodes, the performance of collaborative $Q$-learning is close to random search. In most situations, collaborative $Q$-learning outperforms random search.

GA is inspired by natural science to simulate the evolution of biological populations. After the initialization of population, evolution, selection, mutation, and crossover operations are implemented on the population to produce a new generation that is expected to be closer to the global optimal solution. The evolution continues until a certain step ends or some criteria are met. Algorithm steps include as follows:

1. Select the initial population.
2. Evaluate the fitness of individuals in the population.
3. Select proportionally to produce the next population.
4. Change the population.
5. Judge the stop condition and return to the second step if not satisfied.

Our initial population of individuals as shown in Table 4 is chosen randomly. Individual fitness of the population is the number of packets sent to the BS successfully. The next generation is selected proportionally. Individuals with high adaptability (a bigger number of packets transmitted) will be selected in a higher proportion, and other individuals with low adaptability will be selected at a lower proportion. In this way, a new population is generated. Crossover and mutation are also applied to modify the individuals and generate new individuals. The simulation is implemented using the GA package for Python, named *geneticalgorithm*.

Table 4. Possible solution.

| Node | T1 | T2 | T3 | T4 | T5 | T240 |
|------|----|----|----|----|----|------|
| 1    | 12 | 8  | 10 | 10 | 16 | ...  |
| 2    | 6  | 8  | 18 | 20 | 22 | ...  |
| 3    | 14 | 20 | 14 | 24 | 16 | ...  |
| ...  | ...| ...| ...| ...| ...| ...  |
| 60   | 18 | 14 | 22 | 16 | 8  | ...  |

Table 5. Comparison with random search.

| Number of nodes | Random search | Collaborative Q-learning |
|-----------------|---------------|-------------------------|
|                 | Average | Best | Average | Best |
| 20              | 60,387  | 63,606 | 76,534  |
| 40              | 122,002 | 123,444 | 122,334 |
| 60              | 188,708 | 192,717 | 211,123 |

Table 6. Genetic algorithm performance.

| Number of nodes | Iteration | Population | Best |
|-----------------|-----------|------------|------|
| 20              | 200       | 50         | 58,430|
| 20              | 500       | 40         | 58,900|
| 40              | 100       | 100        | 112,444|
| 40              | 500       | 40         | 113,490|
| 60              | 100       | 100        | 165,834|
| 60              | 200       | 50         | 168,340|
| 60              | 500       | 40         | 168,780|

Adaption to parameter change

Adaption to battery degradation. In order to simulate the battery degradation of a sensor node, the initial energy of the node battery is reduced to 80%, 60%,
and 40% of the original battery capacity. The cooperative Q-learning and SARSA(λ) results are shown in Figures 10 and 11. In this case, the cooperative Q-learning and cooperative SARSA(λ) strategy have similar behavior and can still adapt to changes, and their network throughput performance is better than static transmission by 16.6%–30.1%. In terms of the number of dead nodes, there are 60 nodes in the network. If one node in each round is dead in one moment, the number of dead nodes is counted as 1. There are 240 rounds in each iteration (24 h). In Figure 9, the number of dead nodes refers to the number of all dead nodes in 240 rounds (24 h) which are between 1500 and 3000. Cooperative Q-learning and cooperative SARSA(λ) have fewer dead nodes in every 24 h than the static method by 21.0%–45.9%.

Adaptation to changes in equipment parameters. The sensor nodes in IoT need to experience changes in equipment parameters, such as equipment energy efficiency. This experiment is designed for the change of node energy efficiency. In order to simulate the change of device parameters, the power consumption of the node is changed by 125%, 100%, 75%, 50%, and 25%. The results of the network performance under different equipment parameters are shown in Figures 12 and 13. The results reveal that cooperative Q-learning and cooperative SARSA(λ) are still adaptable and have better performance than the tuned static algorithm. When the original power consumption is increased to 125%, the network throughput rate in all cases is reduced, but the information transmission rate is still higher than that of the static method. At 6.8% and 11.8%, when the power consumption is reduced to 75%, 50%, and 25%, cooperative Q-learning and cooperative SARSA(λ) methods are still higher than the static method by 5.5%–18.5%. In terms of the number of dead nodes, 60 nodes are randomly distributed in the field. There are 240 rounds of data transmission in each iteration (24 h) and the number of dead nodes in each round is recorded as 1. The number of dead nodes refers to the number of all dead nodes in 240 rounds. From Figure 13, cooperative Q-learning and cooperative SARSA(λ) have fewer dead nodes in each iteration than the static method by 8.9%–37.3%, especially in the two high energy consumption situations of 125% and 100%. When the energy consumption is 100%, cooperative Q-learning and cooperative SARSA(λ) methods are 29.7% and 27.0% higher than the static method, respectively. When the energy consumption is 125%, cooperative Q-learning and cooperative SARSA(λ) are respectively higher than the static method 37.3% and 34.6%.

Adaptation to changes in different field node densities. Finally, different densities of network sensor deployment are considered in the field. The density of sensor nodes may affect the clustering and the transmission distance between the nodes. Figure 14 shows the comparison results of constant transmission rate policy, cooperative Q-learning, and cooperative SARSA(λ) when the sensor nodes in a field are 20, 40, and 60, respectively. The cooperative
Q-learning and SARSA(λ) algorithms are averagely better than the constant transmission rate policy in the network throughput.

**Conclusion**

The adaptive power management system of this article uses cooperative Q-learning and SARSA(λ) algorithm to optimize the data transmission rate of EH-WSNs, rather than relying on the manual configuration of the node transmission rate. This is the first application of CRL to adaptively manage the transmission rate of nodes in clustered solar-powered WSNs. Experiments show that the network with CRL power management policy can adapt itself to environmental changes, and sensor nodes can determine the data transmission rate according to their own residual electricity and collectible energy. Compared with the general network, the data transmission performance of the optimized network has been improved. At the same time, the network can adapt to the changes of equipment parameters, battery degradation, and node density changes, which ensures the reliability and stability of networks in the application and enhances the practical application performance of the network. Although this article focuses on maximizing information transmission, these methods are applicable to various situations of WSNs and a wider range of IoT applications by designing new reward function.

The complexity and convergence of the proposed algorithms are analyzed. Algorithms such as cooperative Q-learning and SARSA based on discrete table values have a limit on the number of states which are not suitable for continuous state modeling. Future work will explore using function approximation to learn to estimate the function of $V(s)$ or $Q(s, a)$. When the number of observations is large enough, the linear approximation can converge to the local optimum. However, the linear function approximator is insufficient for capturing complex environments. A large number of nonlinear approximators have been studied in recent years, especially neural networks, such as the Deep Q-Network (DQN) algorithm. The DQN algorithm uses a convolutional neural network (CNN) and the replay memory buffer to obtain a random mini-batch to train the approximator. The function approximation suitable for the energy management environment in WSNs will
be explored while ensuring certain complexity constraints and convergence performance. Experiments will also explore the efficiency of these algorithms by different reward functions to identify how these variables affect the energy management of the nodes. The development of IoT protocols and prototyping systems for IoT environmental monitoring in industries, such as the aquaculture industry, is helpful for further verification and evaluation for the usability and effectiveness of the energy optimization method proposed in this project.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partly funded by the Natural Science Foundation of Zhejiang Province (nos LGF20G010004 and No. LY19F020003).

ORCID iD
Yujia Ge https://orcid.org/0000-0003-4034-0838

References
1. Sudevalayam S and Kulkarni P. Energy harvesting sensor nodes: survey and implications. IEEE Commun Surv Tutor 2011; 13(3): 443–461.
2. Adu-Manu KS, Adam N, Tapparello C, et al. Energy-harvesting wireless sensor networks (EH-WSNs): a review. ACM T Sensor Network 2018; 14(2): 10.
3. Kansal A, Hsu J, Srivastava M, et al. Harvesting aware power management for sensor networks. In: Proceedings of the 43rd ACM design automation conference (DAC), San Francisco, CA, 24–28 July 2006, pp.651–656. New York: IEEE.
4. Zareei M, Vargas-Rojales C, Anisi MH, et al. Enhancing the performance of energy harvesting sensor networks for environmental monitoring applications. Energies 2019; 12: 2794.
5. Zareei M, Vargas-Rojales C, Villalpando-Hernandez R, et al. The effects of an Adaptive and Distributed Transmission Power Control on the performance of energy harvesting sensor networks. Comput Netw 2018; 137: 69–82.
6. Babayo AA, Anisi MH and Ali I. A review on energy management schemes in energy harvesting wireless sensor networks. Renew Sust Energ Rev 2017; 76: 1176–1184.
7. Sutton RS and Barto AG. Reinforcement learning: an introduction. 2nd ed. Cambridge, MA: The MIT Press, 2018.
8. Shresthamali S, Kondo M and Nakamura H. Adaptive power management in solar energy harvesting sensor node using reinforcement learning. ACM T Embed Comput Syst 2017; 16: 181.
9. Heinzelman WR, Chandrakasan AP and Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd annual Hawaii international conference on system sciences, Maui, HI, 7 January 2000, pp.10–19. New York: IEEE.
10. Wei D, Jin Y, Vural S, et al. An energy-efficient clustering solution for wireless sensor networks. IEEE T Wirel Commun 2011; 10(11): 3973–3983.
11. Chen G, Li C, Ye M, et al. An unequal cluster-based routing protocol in wireless sensor networks. Wirel Netw 2009; 15(2): 193–207.
12. Peng S and Low CP. Energy neutral routing for energy harvesting wireless sensor networks. Ad Hoc Netw 2015; 28: 1–16.
13. Wang T, Heinzelman W and Seyedi A. Maximization of data gathering in clustered wireless sensor networks. In: Proceedings of the 2010 IEEE global telecommunications conference GLOBECOM 2010, Miami, FL, 6–10 December 2010, pp.1–5. New York: IEEE.
14. Mehrabi A and Kim K. General framework for network throughput maximization in sink-based energy harvesting wireless sensor networks. IEEE T Mobile Comput 2017; 16(7): 1881–1896.
15. Ge Y, Nan Y and Chen Y. Maximizing information transmission for energy harvesting sensor networks by an uneven clustering protocol and energy management. KSII T Internet Inf Sci 2020; 14(4): 1419–1436.
16. Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms, 2017, https://arxiv.org/abs/1707.06347 (accessed 20 August 2020).
17. Ortiz A, Al-Shatri H, Li X, et al. Reinforcement learning for energy harvesting decode-and-forward two-hop communications. IEEE T Green Commun Netw 2017; 1(3): 309–319.
18. Aoudia FA, Gautier M and Berder O. RLMan: an energy manager based on reinforcement learning for energy harvesting wireless sensor networks. IEEE T Green Commun Netw 2018; 2(2): 408–417.
19. Hsu RC, Liu CT and Lee WM. Reinforcement learning-based dynamic power management for energy harvesting wireless sensor network. In: Proceedings of the international conference on industrial, engineering and other applications of applied intelligent systems (Lecture notes in computer science), Graz, 9–11 July 2009, vol. 5579. Berlin; Heidelberg: Springer.
20. Hsu RC, Liu CT and Wang HL. A reinforcement learning-based ToD provisioning dynamic power management for sustainable operation of energy harvesting wireless sensor node. IEEE T Emerg Top Comput 2014; 2(2): 181–191.
21. Rioual Y, Moullec YL, Laurent J, et al. Reward function evaluation in a reinforcement learning approach for energy management. In: Proceedings of the biennial Baltic Electronics conference, Tallinn, 8–10 October 2018, pp.1–4. New York: IEEE.
22. Fraternali F, Balaji B and Gupta R. Scaling configuration of energy harvesting sensors with reinforcement learning. In: Proceedings of the 6th international workshop on energy harvesting & energy-neutral sensing systems,
23. Bernstein DS, Givan R, Immerman N, et al. The complexity of decentralized control of Markov decision processes. *Math Oper Res* 2002; 27(4): 819–840.

24. Foerster J, Nardelli N, Farquhar G, et al. Stabilising experience replay for deep multi-agent reinforcement learning. In: *Proceedings of the 34th international conference on machine learning*, Sydney, NSW, Australia, 6–11 August 2017, pp.1146–1155. New York: ACM.

25. Tan M. Multi-agent reinforcement learning: independent vs. cooperative agents. In: *Proceedings of the 10th international conference on machine learning*, Amherst, MA, 27–29 June 1993, pp.330–337. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

26. Matignon L, Laurent GJ and Fort-Piat NL. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowl Eng Rev* 2012; 27(1): 1–31.

27. Claus C and Boutilier C. The dynamics of reinforcement learning in cooperative multiagent systems. In: *Proceedings of the 15th national conference on artificial intelligence*, Madison, WI, 26–30 July 1998, pp.746–752. Menlo Park, CA: AAAI.

28. Littman ML. Value-function reinforcement learning in Markov games. *Cogn Syst Res* 2001; 2(1): 55–66.

29. Hu J and Wellman MP. Nash Q-learning for general-sum stochastic games. *J Mach Learn Res* 2003; 4: 1039–1069.

30. Littman ML. Friend-or-Foe Q-learning in general-sum games. In: *Proceedings of the 18th international conference on Machine Learning*, Williamstown, MA, 28 June–1 July 2001, pp.322–328. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

31. Khan MI and Rinner B. Energy-aware task scheduling in wireless sensor networks based on cooperative reinforcement learning. In: *Proceedings of the IEEE international conference on communications workshops (ICCW)*, Sydney, NSW, Australia, 10–14 June 2014, pp.871–877. New York: IEEE.

32. Emre M, Gü r G, Bayhan S, et al. CooperativeQ: energy-efficient channel access based on cooperative reinforcement learning. In: *Proceedings of the IEEE international conference on communication workshop (ICCW)*, London, 8–12 June 2015, pp.2799–2805. New York: IEEE.

33. Wu Y, Liu W and Liu Y. Cooperative reinforcement learning based through optimization in energy harvesting-wireless sensor network. *Int J Innov Comput I* 2018; 14(6): 1993–2010.

34. Ge Y, Nan Y and Bai J. A hybrid prediction model for solar radiation based on long short-term memory, empirical mode decomposition, and solar profiles for energy harvesting wireless sensor networks. *Energies* 2019; 12(24): 4762.

35. Watkins CJCH. *Learning from delayed rewards*. London: King’s College London, 1989.

36. Singh S, Jaakkola T, Littman ML, et al. Convergence results for single-step on-policy reinforcement-learning algorithms. *Mach Learn* 2000; 38: 287–308.

37. Puterman ML. *Markov decision processes: discrete stochastic dynamic programming*. Hoboken, NJ: Wiley-Interscience, 2005.

38. Lauer M and Riedmiller M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: *Proceedings of the 17th international conference on machine learning*, San Francisco, CA, 29 June–2 July 2000, pp.535–542. San Francisco, CA: Morgan Kaufmann Publishers, Inc.

39. US Department of Energy. National solar radiation database, http://rredc.nrel.gov/solar/old_data/nsrdb/ (accessed 20 August 2020).

40. Koyi ON, Yang HS and Kwon Y. Impact of base station location on wireless sensor networks. In: *Proceedings of the 4th computer science On-line conference 2015 (CSOC2015): intelligent systems in cybernetics and automation theory*, Zlin, 27–30 April 2015, pp.151–162. Cham: Springer.

41. Karray F, J mal MW, Garcia-Ortiz A, et al. A comprehensive survey on wireless sensor node hardware platforms. *Comput Netw* 2018; 144: 89–110.

42. Solgi RM. *geneticalgorithm 1.0.2* (Genetic algorithm python library), 2020, https://pypi.Org/project/genetical-gorithm/ (accessed 2 February 2021).

43. Sutton RS, McAllester DA, Singh SP, et al. Policy gradient methods for reinforcement learning with function approximation. *Adv Neur In* 2000; 12: 1057–1063.

44. Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. *Nature* 2015; 518(7540): 529–533.