Soft Computing
https://doi.org/10.1007/s00500-023-08734-4

FOCUS

Energy-balanced routing in wireless sensor networks with reinforcement learning using greedy action chains

Zhibin Liu[1] · Xinshui Wang[1]

Accepted: 30 May 2023

Abstract
The challenge of routing in energy-constrained wireless sensor networks is to balance energy and extend network lifetime, and reinforcement learning is an effective method to address this problem. However, poor learning and decision-making efficiency are still the main factors affecting the network routing performance. In this research, RLR-TET is proposed as a reinforcement learning approach that evaluates the shortest routing and achieves node energy balance. First, this study employs a method in which each node broadcasts control information directly to its neighbor nodes, allowing learning to be accomplished by simple communication. Second, multiple greedy action chains form a tree structure by learning to build a greedy action chain. Third, the tree is dynamically adjusted during the learning process. Toward the root, each sensor node can obtain its optimal next hop; toward the leaf, each sensor node can achieve the update of the learning errors. Fourth, the learning errors of *V*-values are backpropagated along the tree to multi-layer nodes using the eligibility traces approach, achieving fast multi-step updates. In our experiments, we compared the method of this paper with five other algorithms. The number of packets delivered, the number of alive nodes, and the standard deviation of the nodes' remaining energy all improved significantly over time. Experiments have shown that our method responds quickly to network changes, improves the energy balance of network nodes, and successfully extends the lifetime of the network.

Keywords Wireless sensor networks · Routing · Reinforcement learning · Energy balance · Greedy action chains · Error tree-based backpropagation · Eligibility traces

1 Introduction

The wireless sensor network is a self-organized network with a particular topology that comprises fixed or mobile sensing nodes that monitor different types of environmental data such as temperature, pressure, light intensity, humidity, and noise (Al-Janabi et al. 2021). These small, low-cost nodes have some computing and processing power. Wireless sensors are built self-organizing, and the collected information is sent to base stations or sinks via multi-hop relaying. Developing wireless sensor network technology aims to link more intelligent devices and provide more accessible, dependable, and rapid services. Improving network throughput, reducing and balancing load power consumption, increasing network response speed, and extending network lifetime have been hot research topics.

The routing problem is fundamental to the study of wireless sensor networks (Saleh et al. 2015). In contrast to fixed communication and computer networks, wireless sensor networks are more sophisticated in multi-hop routing, manifested in the following aspects. First, wireless sensor network nodes are powered by batteries, but node energy is limited, necessitating optimization of routing, load balancing, and reducing the number of routing hops to extend its life (Mohammed and Al-Janabi 2022). Second, changes in network architecture are readily induced by battery depletion, instability of communications connection signals, or signal interference (Srivastava and Mishra 2022). Third, if the data transmission path is not balanced, data cannot be transferred in time and becomes

✉ Zhibin Liu
lzbxian@qfnu.edu.cn

Xinshui Wang
wxinshui@126.com

[1] School of Computer Science, Qufu Normal University, Rizhao, Shandong, China

Published online: 21 June 2023

overstocked in the node buffer, resulting in packet loss, time delay, and performance deterioration. Fourth, changes in network architecture are caused by battery power, congestion, and sensor node movement. These issues have posed significant challenges to network routing (Lin et al. 2020). A pre-defined routing strategy decides the traditional routing scheme. However, this is only relevant to a given network environment and cannot be flexibly optimized and modified in a changing environment. Traditional approaches based on mathematical models are likewise challenging to implement. It is difficult to describe the changing network topology with an accurate mathematical model, and even if it can, the model is complicated, computationally costly, and has the drawback of poor self-adaptability.

In recent years, machine learning (Kadhuim and Al-Janabi 2023) approaches have gained much interest for routing in a continually changing network to provide self-adaptive solutions to wireless sensor network issues with changing topologies (Yang 2020). Among these, reinforcement learning (Sutton and Barto 1998) is a significant field of machine learning with significant application potential in wireless sensor networks (Frikha et al. 2021; Kaur et al. 2021; Cho and Lee 2020; Kwon et al. 2020). Perception is used in reinforcement learning to make decisions (Zhang et al. 2020). Through continual trial and error, the agent enhances the value of its experience and progressively optimizes its approach. Sensor nodes can continually identify the best option for adaptive routing. Reinforcement learning has emerged as a fundamental approach for determining the best pathways to sinks in wireless sensor networks (Yau et al. 2015), but its optimal path is shifting. We must consider the network's QoS indicators, such as the degree of network congestion, the energy consumption of sensor nodes, the amount of node buffer space, and node mobility. Many methods consider the shortest path routing of wireless sensor networks and these dynamic QoS indicators (Gazi et al. 2021). It is simple to understand that the reinforcement learning process must keep up with the network's constant changes to develop improved routing solutions. For network routing, out-of-date routing strategies are worthless. As shown in a vast quantity of literature, the routing algorithm based on reinforcement learning completely incorporates the network's QoS metrics but seldom considers improving the learning rate (Mammeri 2019, 2019).

Boyan and Littman (1993) pioneered the use of reinforcement learning in wireless sensor routing with their Q-routing method. The Q-routing method combines active and on-demand routing protocols (Zaraket et al. 2021) to maintain a Q-table through information exchange. Furthermore, it discovers the best information transmission method using exploitation and exploration.

Maintain the Q table, which has two update methods: periodic and triggered. The periodic update interval is set to around 10 s. Thus, if 10 routing nodes are traversed from the source node to the destination node, the transmission time will be 110 s. This technique fails to provide an efficient decision process because its learning convergence rate is far from keeping up with the changing topology of the sensor network. If learning efficiency is pursued by drastically lowering periodic update intervals, network information congestion and node energy consumption will occur. The update time between nodes is a few milliseconds with triggered updates, and only individual nodes are accessed. Sensor networks generally utilize less energy and are less likely to generate information package congestion. This approach, however, can only reverse one update step, which is still wasteful. Jiang et al. (2019) used the Sarsa(λ) method for wireless sensor communication, leveraging eligibility traces and multi-step backpropagation of errors to enhance convergence speed and learning efficiency. Like the Q-routing approach, the Sarsa(λ) algorithm ignores node power consumption and balance, causing the network's lifetime to end prematurely. At the same time, this method uses linear error backward updating, which has less learning efficiency and adaptive ability than the method proposed in this paper.

This study presents a reinforcement learning routing protocol using tree-based eligibility traces (RLR-TET). During learning, greedy action chains are created, and the various greedy action chains form a tree structure. The network uses this tree to keep the routing decisions of the wireless network so that the system may be updated swiftly within a specific time interval. The large-scale wireless sensor network establishes a multi-hop network transmission mechanism. Reinforcement learning is incorporated into network communication, and its reward function primarily incorporates node relay, communication energy consumption between nodes, and node residual energy. A multi-agent learning mechanism accomplishes the network's energy and data load balancing. The network's functioning is a dynamic process in which time and energy consumption are critical. If the learning efficiency is too low, the learning process will not always converge in a dynamic environment, leading the system to operate in a non-optimized environment with unoptimized routing performance. The RLR-TET approach has substantially enhanced the overall performance of wireless sensor networks. The following are the primary contributions of this study.

(1) Nodes broadcast low-power, small-data control packets to neighboring nodes at reduced intervals, and sensor network nodes use reinforcement learning to create a tree consisting of greedy action chains automatically. During the learning process, the tree's structure can be dynamically adjusted.
(2) In the tree, toward the root, each sensor node can obtain its optimal next hop; toward the leaf, each sensor node can achieve the update of the learning errors.
(3) During learning, V-value learning errors are back-propagated along the tree to multi-layer nodes using the eligibility traces approach, resulting in multi-step updates.

During the learning process, the network nodes implicitly construct an adaptive tree. This tree can achieve excellent learning efficiency and decision performance, giving it a significant advantage over other algorithms.

The main innovations in this paper compared to other reinforcement learning based sensor routing algorithms are described as follows. Previous sensor routing algorithms can only back-propagate a node's Q-value update by one step when performing reinforcement learning. Even if overhearing techniques are used, they can only influence the Q updates of the node's neighbors. In this paper, the idea of greedy action chains is proposed and a dynamically changing tree of greedy action chains is constructed. Unlike previous algorithms, our approach propagates updates from the root to multiple nodes in the tree. This greatly improves the learning efficiency of the sensor networks and results in more optimal and trustworthy routing decisions.

The remainder of this work is organized as follows.

The second section discusses the use of reinforcement learning in wireless sensor networks. The third section explains the reinforcement learning method and the issue to be solved in this study. The fourth section discusses the notion and algorithm of RLR-TET and the creation of greedy action chains in wireless sensor networks with their tree. The fifth section presents simulation results for the RLR-TET method and compares them to previous algorithms. This paper is summarized in the sixth section.

2 Related work

Traditional self-organizing network routing methods are classified into three forms based on the path discovery method: active protocols, reactive routing protocols, and hybrid routing protocols. A reactive routing protocol is DSR (Dynamic Source Routing Protocol) (Johnson and Maltz 1996). When the source node wishes to transmit data to the target node, it broadcasts a request packet to move information via network nodes until the target node receives it or the information reaches the node that links to the target node. The node then sends the routing information back to the source node, which generates a routing path. A multi-hop reactive routing protocol is AODV (Ad Hoc On-demand Distance Vector) (Das et al. 2003). This method, like the DSR approach, uses a route discovery mechanism. The network's nodes only keep a routing table for the relevant nodes. The routing database records the target node's next-hop address and the number of hops to the destination node and utilizes the sequence number to identify the most recent routing results.

Because reinforcement learning is independent of the model, it is frequently researched in wireless sensor networks with frequent topology changes (Mammeri 2019; Guo and Zhang 2014). The Q-routing method (Boyan and Littman 1993) considers the quickest path and the degree of data congestion during data transfer. By learning, the algorithm is more adaptable than the shortest path routing approach and can avoid congested nodes. However, neither the node's mobility nor its power consumption is taken into consideration by this method. This approach can converge correctly when the network topology changes slowly, but it is ineffective when it changes fast. Oddi et al. (2014) improved the Q-routing algorithm. The algorithm uses residual energy to balance the routing of sensors while optimizing the control overhead of sensor nodes, thereby increasing the lifetime of the network.

The DFES-AODV algorithm (Chettibi and Chikhi 2016) is an AODV routing protocol that employs fuzzy energy states to fuzzily calculate a single node's power and energy usage. Unlike previous fuzzy logic-based approaches, the membership function of the input state value in this method is dynamic, and it is adaptively altered throughout the network's working process to obtain higher generalization performance. This approach integrates fuzzy logic and reinforcement learning into wireless network routing, boosting network performance while balancing battery usage.

The AdaR (Wang and Wang 2006) method optimizes the goal by considering path length, load balancing, communication connection dependability, and data aggregation. The node sends the samples to the base station through the routing node, and the base station learns from a vast number of sample sets. The learning technique then uses the Least-Squares Policy Iteration (LSPI) method to train the samples toward the best strategy gradually. After the base station determines the best method, it broadcasts the information to each sensor node, allowing them to send data more efficiently. This approach uses more samples than table-based reinforcement learning algorithms and converges faster. It is appropriate for cases in which network QoS indicators fluctuate often.

Hu and Fei (2010) presented the QELAR algorithm based on *Q* learning and created a network routing approach that extends the network's lifetime by attaining energy balance. It computes the reward value depending on the node's remaining energy and the energy distribution of the surrounding nodes. The QELAR method employs an underwater sensor network as the application scenario; each node only retains the information of surrounding nodes, and communication is confined to local sections, reducing system overhead. With unstable transmission links and moving nodes, this method can swiftly adapt to changes in network architecture. The sensor network nodes' power is restricted. If the shortest path and the fewest amount of communication hops are chosen as ideal goals, the power of the nodes on these pathways will be quickly depleted, limiting the network's life. The QELAR algorithm's reward value incorporates the nodes' remaining energy and energy distribution, resulting in a more balanced network energy consumption, a longer network life, and a network topology more responsive to changes in the network topology.

Maleki et al. (2017) used model-based reinforcement learning approaches to solve the routing problem in mobile ad hoc networks. Nodes may receive energy from their surroundings, causing energy acquisition and consumption to fluctuate randomly over time. Furthermore, transmission delay will be caused by the network's link quality. This approach employs reinforcement learning to achieve intelligent routing, intending to reduce route transmission time and node energy loss. The Markov decision process realizes routing integrating network mobility and energy dynamics without prior knowledge. The model-based learning process is accomplished using window statistics, and the learning rate has been substantially enhanced.

Zheng et al. (2018) used reinforcement learning for the intelligent data transmission of a UAV (Unmanned Aerial Vehicle) and suggested a directed MAC protocol based on location prediction. The protocol is divided into three stages: location prediction, communication control, and data transfer. The approach of integrating antenna orientation and position prediction is accomplished. The UAV anticipates its position and speed, updates its local strategy, and accomplishes the routing strategy's automated development and a high degree of autonomy with the help of the reinforcement learning algorithm. The proposed algorithm has strong reliability and resilience, and it can generate communication links quickly in a high-speed environment network and transmit data with minimal latency.

Basagni et al. (2019) achieved high-performance underwater sensor node transmission and routing. The data flow is classified as either steady or urgent. A reinforcement learning framework is used to build the multi-modal approach MARLIN-Q, and the optimal route is chosen based on the data's multiple QoS indicators. It offers improved adaptive performance when network scale and critical data transfer changes. It has low latency, low power consumption, and high dependability.

Wang et al. (2020) developed a distributed adaptive routing algorithm (EDACR) for heterogeneous wireless multimedia sensor networks, as well as a routing method from the source node to the sink with transmission latency and energy limitations. Nodes in heterogeneous networks transmit text and multimedia data, such as photographs. The paper categorizes the node energy level and addresses each one independently. The optimization path is continuously changed through reinforcement learning based on the node's remaining energy, transmission reliability, and communication delay, with the periodic update of network knowledge and structure to ensure the balanced distribution of various QoS indicators and energy.

The network is very prone to congestion due to the enormous transfer of data packets. Ding et al. (2019) presented two deep reinforcement learning approaches that employ a centralized method to learn based on historical experience to address network routing problems, minimize network congestion, and shorten the path length of data packet transmission. The SDMT-DQN approach, for example, utilizes the pair of source-target nodes as a sample to create a neural network, whereas the DOMT-DQN method uses the target node as a sample to build a neural network. Kaur et al. (2021) also used a deep learning approach for intelligent decision-making, which divides the entire network into unequal clusters based on the current data load of the sensor nodes to prevent the network from dying prematurely.

The QGrid method (Li et al. 2014) separates the operational area of cars into grids in which all vehicles are in motion, intending to improve vehicle communication and the route across the grids via reinforcement learning. The QGrid method uses Q-learning to compute the Q-value needed to reach the nearby grids, while the vehicle picks the best next grid by querying the Q table. Compared to the GSPR (Karp and Kung 2000) method, this method improves the delivery ratio, hop count, delay, number of forwards, and other metrics.

The QGeo algorithm (Jung et al. 2017) provides a reinforcement learning-based geographic routing strategy. In cases with high mobility and a lack of global knowledge, UAVs execute learning and policy optimization in a distributed computing fashion, resulting in reduced network traffic for QGeo during network performance evaluation. The QGeo method has three modules: position estimation, neighbor table management, and Q-learning. During communication, nodes learn by receiving HELLO messages regularly and learning knowledge from neighboring nodes. Based on the QGeo algorithm, the RFLQGeo algorithm (Jin et al. 2019) develops a routing strategy to increase network performance using a reverse reinforcement learning method, and the reward function of this scheme is created by real-time learning. The approach features a high packet delivery rate, a low number of re-transmissions, and a low end-to-end latency.

Renold and Chandrakala (2017) introduced an MRL-SCSO algorithm for network topology management and data distribution that employs multi-agent reinforcement learning and energy-aware convex packet algorithms to perform autonomous design and optimization of adaptive wireless sensor networks. The MRL-SCSO method maintains a stable network topology, picks influential neighbor nodes via a learning process, and sets network boundaries using a convex packet algorithm to sustain wireless network connection and coverage under heavy traffic situations.

Guo et al. (2019) developed an RLBR approach for maximizing the lifetime of wireless sensor networks using reinforcement learning routing algorithms. The reward function considers the link distance, the node's remaining energy, and the number of hops to the sinks, and the node receives feedback from its nearby nodes after transmitting a packet. The transmit power of the nodes is dynamically modified so that the sensor nodes retain a stronger connection with the sinks, and this strategy equalizes the sensor network's energy consumption, decreases total energy consumption, and increases packet transmission efficiency. Bouzid et al. (2020) advanced on the RLBR method by presenting the R2LTO algorithm, which also attempts to optimize network lifetime and energy consumption by postponing network node death with extended learning time and dynamic path selection. When determining the reward function, the direct distance of the node, the node's remaining energy, and the number of hops from the node to the sink node are all taken into account. The incentive value was modified by the authors depending on the ratio of the node's transmission energy to the maximum energy consumption.

Li et al. (2020) presented a multi-agent reinforcement learning routing protocol DMARL for underwater optical wireless sensor networks, characterizing the network as a distributed multi-agent system. The reinforcement learning algorithm uses the nodes' residual energy and the connections' quality as reward values to adapt to changes in topology to prolong the network's lifetime. Two tactics are used to accelerate learning convergence: one is to utilize node location to initialize the Q value, and the other is to alter the learning rate parameter to adapt to the constantly changing network rapidly. Serhani et al. (2020) proposed AQRouting, an adaptive routing protocol that can assess the movement level of nodes at different times and learn the movement status inside the network. Nodes modify their routing behavior based on the network circumstances around them, which can increase connection reliability and packet transmission rate in both static and dynamic mobile environments.

Table 1 compares the 3 properties of the 11 algorithms.

3 System modeling and reinforcement learning

3.1 System model and problem description

In a wireless sensor network, nodes are considered nonrechargeable; for comparative reasons, all nodes are assumed to have the same starting power level. Many sensor nodes are equally dispersed in a defined region, and the network has one or more sinks. The sinks are expected to have adequate processing and electrical power and are linked to the central server through a high-speed transmission channel. If a sensor node creates or receives a packet that cannot be transmitted directly to the sinks, the packet can only be relayed by other nodes and sent to the

Table 1 Comparison of various algorithms

Routing protocols	Learning entities	Consider energy balance	Update method
Q-routing	Discrete	No	Single-step update
AdaR	Centralized	No	Single-step update
RLBR	Discrete	Yes	Single-step update
Sarsa(λ)	Discrete	No	Single-step update
Sarsa(λ)-EB	Discrete	Yes	Single-step update
MRL-SCSO	Discrete	Yes	Single-step update
SDMT-DQN	Centralized	No	Single-step update
EDACR	Discrete	Yes	Single-step update
R2LTO	Discrete	Yes	Single-step update
OPT-EQ-Routing	Discrete	Yes	Single-step update
RLR-TET	Discrete	Yes	Tree multi-step update

sinks in a multi-hop fashion (Guo et al. 2019). As a result, specific sensor nodes are utilized more frequently than others, resulting in uneven energy consumption in the network. Individual nodes with a high frequency of usage, whose energy is quickly drained and cannot continue to perform transfer tasks, will cause the entire network to fail to work correctly. The primary strategy for solving this problem in wireless sensor networks is the balanced usage and scheduling of node energy. A network should also consider QoS measures such as node mobility, link channel allocation, and communication quality. In this work, just the number of hops from nodes to sinks and node power are evaluated to show and validate the algorithm's effectiveness.

One data transmission is handled as a Markov decision process in a wireless sensor network environment, and one hop data transfer is treated as an agent's state change with feedback reward values. The following performance factors must be considered.

(1) If the power of the data sensor node is adequate, bigger feedback is supplied to encourage the sending node to communicate the data; if the power is insufficient, smaller feedback is returned to discourage the sending node from transmitting the data. If the sensor node power is imbalanced, the network will break, and the entire life will expire prematurely.

(2) When a sensor node sends data, if the receiving node is close to the sending node, only a lower transmit power is required to send the packet, resulting in a higher reward value; if the receiving node is far from the sending node, a higher transmit power is required to send the packet, resulting in a lower reward value. This strategy promotes data transmission in an energy-efficient manner while simultaneously reducing interlink interference.

(3) Short pathways are critical in routing wireless sensor networks, whereas node power balancing and load balancing are key auxiliary metrics. If the number of hops of the routing trajectory is used as a measure of the routing length, the more hops, the greater the network's instability and energy consumption; however, if only the shortest path is considered, the energy of the nodes on a specific shortest path trajectory is quickly exhausted, causing severe congestion.

(4) To show the approach's usefulness in this study, a plane routing protocol is used, and it is assumed that each sensor node's sensing radius is constant, while nodes beyond the sensing radius cannot interact with one another. The node cannot acquire global information from the sensor network, and the communication channel between the nodes is symmetrical.

(5) By receiving packets, each sensor node is aware of the position of its neighbors.

3.2 Reinforcement learning and its mathematical representation

Wireless sensor network nodes provide two functions: one is to generate data, and the other is to act as packet routing relays. We consider wireless sensor network data transmission a multi-agent reinforcement learning issue. Each data packet or message is considered an agent, and complicated reinforcement learning is performed through information transmission and feedback.

Reinforcement learning is a subfield of machine learning in which the goal is to continually learn and optimize the state-to-action mapping by interacting with the environment and computing the cumulative value of the reward earned from each action. Q-learning (Watkins and Dayan 1992) is a model-free reinforcement learning approach theoretically based on the Markov decision process (MDP). Assuming the model is known, the dynamic programming approach is utilized to solve it. However, this type of dynamic model is unavailable in the context of a wireless sensor network. With a high number of samples in the transmission process, the optimization action that approximates dynamic programming is gradually discovered in the learning process. At time t, the agent is in the state s_t, and the action that the agent can take at this time is $a_t \in A(s_t)$. After executing the action a_t, the random reward value that the environment feeds back to the agent is r_t the average reward value R_{s_t}, and the probability of transitioning from state s_t to state s_{t+1} is: $P_{s_t,s_{t+1}}(a_t)$.The Q value is defined as follows:

$$Q^{\pi}(s_t,a_t) = R_{s_t}(a_t) + \gamma \sum\nolimits_{s_{t+1}\in S_{t+1}} P_{s_t,s_{t+1}}(a_t) \sum\nolimits_{a_{t+1}\in A_{t+1}} \pi[a_{t+1}|s_{t+1}] Q^{\pi}(s_{t+1},a_{t+1}) \tag{1}$$

The goal of obtaining the optimal solution is to maximize $Q^{\pi}(s,a)$ by adjusting the strategy $\pi(a|s)$.

Since the environment model is unknown, the Q-learning algorithm solves this problem by maximizing each state-action pair with constant sampling and optimizing iterations to maximize the policy π^{*} under state s. The iterative formula for calculating the Q-value is as follows:

$$\mathrm{Q}(\mathrm{s}_t,a_t) = (1-\alpha)Q(s_t,a_t) + \alpha\left[r_t + \gamma \max_{a_{t+1}\in A(s_{t+1})} Q(s_{t+1},a_{t+1})\right] \tag{2}$$

When the agent is sampled by action trajectory, the Q-learning technique is used to update the Q-values of the nodes offline, using the ε - greedy approach. The following is how the state s gets its action a:

$$a^{\mathcal{E}-\text{greedy}}(s) = \begin{cases} \max\limits_{a\in A(s)} Q(s,a), \ \text{Prob} = 1-\mathcal{E} \\ \text{random action in}\, A(s), \ \text{Prob} = \mathcal{E} \end{cases} \tag{3}$$

The optimization goal can be reached using the Q learning method if each state-action pair is visited balanced and the number of accesses is large enough. However, because the network optimization process is online, learning efficiency is crucial, as learning results must converge as fast as feasible with as little node contact as possible. Slower learning efficiency can cause data packet congestion, packet loss, and network power imbalance, significantly impacting network performance.

The TD(λ) (multi-step temporal-difference learning based on eligibility traces) algorithm is a better solution, in which the node is updated in a single step within a specified time interval, and its knowledge can be used to update the Q values of multiple other nodes, avoiding Q-learning with only one step update.

The eligibility trace is defined as follows:

$$e_{t+1}(s) = \left\{ \begin{array}{c} \gamma\lambda e_t(s), \ s \neq s_t \\ \gamma\lambda e_t(s) + 1, \ s = s_t \end{array} \right\} \tag{4}$$

$$e_{t+1}(s) = \left\{ \begin{array}{c} \gamma\lambda e_t(s), \ s \neq s_t \\ 1, \ s = s_t \end{array} \right\} \tag{5}$$

Equation (4) employs an accumulation technique; when the node is visited, its eligibility trace is enhanced by one on its original basis. Equation (5) uses the replacement method. When a node is visited, its eligibility trace is set to 1, and the new trace replaces the old one regardless of its previous value. In this paper, the replacement method is used in the subsequent sections. The efficiency of the Q-table update is significantly increased by applying the eligibility trace method to propagate a node's one-step Q-value update backward along the best link direction for many steps. The one-step update of the Q-value is as follows:

$$\delta_t = R_{t+1} + \gamma V(s_{t+1}) - V(s_t) \tag{6}$$

The Q-value update formula for each node on the state-action trajectory is as follows:

$$V(s) = V(s) + \alpha\delta_t e_t(s) \tag{7}$$

4 RLR-TET algorithm performance metrics

Our proposed RLR-TET method is a reinforcement learning wireless routing protocol that employs eligibility traces and back-propagates errors in the tree direction. The algorithm transfers information between nodes and optimizes them through progressive learning to meet the aim of node power balance, which increases the overall network's lifetime. The energy model of node transmission and the number of hops from node to sink are critical parameters during reinforcement learning.

4.1 Energy model

The sensor node is made up of a power supply subsystem, a sensing subsystem, a computer subsystem, and a communication subsystem. Each sensor node has the same initial energy E_{init}, and its remaining energy during use is E_{rem}.

Sensor nodes consume energy in various ways, including sensing environmental information, computing and processing information, transmitting and receiving information, spending energy in an idle state, consuming energy in a sleep state, and other energy-consuming activities. All of them consume the most power, with transmitting data using the most, while receiving data and idle state consume energy but less than sending data, and other scenarios consume minimal energy. The discussions and calculations in this paper use an energy model for analysis, taking only the energy consumption of nodes sending and receiving data into account. In contrast, the energy consumption in the idle state can be avoided as much as possible by optimizing the node sleep time, and the energy consumption in other cases is not considered. The reinforcement learning algorithm is utilized in this research. In practical applications, obtaining an accurate mathematical model of the sensor nodes is not necessary, as it is possible to make the correct decision based on the environmental feedback reward information. Therefore, the algorithm used in this paper does not lose its generality.

The nodes inside sensor node n's communication radius are referred to as its neighbor nodes, and all of them form a set, denoted as Neighbor(n), and this communication radius is written as: R_{neighbor}. When sensor node n delivers a k-bits packet to a nearby node with a distance of d, both the transmitter and the receiver spend energy, as defined by Guo et al. (2019).

$$E_{Tx}(k,d)=\begin{cases} E_{\text{elec}}k+\varepsilon_{\text{fs}}kd^2, d<d_0 \\ E_{\text{elec}}k+\varepsilon_{\text{amp}}kd^4, d\geq d_0 \end{cases} \tag{8}$$

$$\mathrm{E}_{\mathrm{Rx}}(k)=E_{\text{elec}}k \tag{9}$$

where k is the length of the packet sent or received by the node, d is the distance between the sending and receiving nodes,$\mathrm{E}_{\mathrm{Rx}}(k)$ is the power consumed by the node to send a packet of length k bits to the receiving node at the range d, and $\mathrm{E}_{\mathrm{Rx}}(k)$ is the power consumed by the node to receive a packet of length k bits. Constants include m,E_{elec},ε_{fs}, and ε_{amp}.E_{elec} denotes the energy required by the transmitting or receiving circuit to process each bit of data,ε_{fs}, and ε_{amp} denote the energy consumed by the transmitting node to convey 1 bit of data per unit distance by broadcasting a wireless signal, and m is the propagation attenuation index. The distance threshold for the amplifier to change the power is denoted d_0 and computed as follows:

$$d_0=\sqrt{\frac{\varepsilon_{\text{fs}}}{\varepsilon_{\text{amp}}}} \tag{10}$$

4.2 The number of hops between a node and a sink

The nodes transmit packages to the sink, and packets are routed through the fewest hops, the best configuration for data transmission. However, networks have drawbacks when packets are all routed most shortly. Frequent visits to fixed nodes can cause these nodes to die prematurely in the long run, while too many visits to fixed nodes might cause congestion in the near run.

When a node connects with its neighbors, it learns the number of hops between the neighbors and the sink. Iterating repeatedly allows the number of hops from the node to the sink to be obtained faster and then participates in updating the network nodes' Q and V values in the form of reward values. Some nodes will run out of energy and die during network operation, or their locations will be altered for various reasons. Hence it is critical to fix the node hop count throughout the learning process.

4.3 The data structure of node information

Each node needs to maintain a node information table to record its information and the information of neighbor nodes. The entire network maintains a dynamic Q table, and each entry of the Q table is recorded in the node information table to provide necessary data for learning and decision-making. The data structure of node information is illustrated in Table 2.

The traditional approach in the learning process of reinforcement learning-based wireless sensor networks employs four types of packets to communicate between nodes. First, there are probe message packets sent between nodes for reinforcement learning. The second is the probe message packet's reply message. Finally, there are the data packets. Fourth, after transmitting packets to surrounding nodes, their reply messages.

The first and second information interaction procedures are simplified and enhanced in this study: each node broadcasts control information to its neighbor nodes, the adjacent nodes receive the information, and then their Q and V values are updated.

In network communication, the data packet is significantly longer than the control packet, and the network architecture is constantly changing. The data packet will not be delivered to the target node without appropriate learning, resulting in packet loss and increased network congestion and power consumption. Nodes delivering control packets at regular intervals to achieve learning is also a vital way of network optimization; of course, packet transmission also includes learning information to achieve network policy optimization jointly.

5 RLR-TET algorithm description

The RLR-TET algorithm's main components are as follows: the minimum number of hops from nodes to sinks, nodes periodically sending control packets to neighboring nodes to drive network Q table updates, generation of a tree structure consisting of greedy action chains,

Table 2 The data structure of node information

Node item 1	Node item 2	Description
ID		Unique identifier of the node
V		*V* value of the node
coordinateX		X coordinate of the node
coordinateY		Y coordinate of the node
optNextNode		Optimal next-hop node
optLastNodes	optLastNode1 optLastNode2 …	The set of predecessor nodes of this node, nodes in the set, make this node the optimal hop
minHopCount		Minimum number of hops from the current node to the sink
neighborNodes	neighborID1	The ID of neighbor node 1
	neighborQ1	*Q* value of neighbor node 1
	coordinateX1	X coordinate of neighbor node 1
	coordinateY1	X coordinate of neighbor node 1
	neighborID2	The ID of neighbor node 2
	neighborQ2	*Q* value of neighbor node 2
	coordinateX2	X coordinate of neighbor node 2
	coordinateY2	X coordinate of neighbor node 2
	…	…

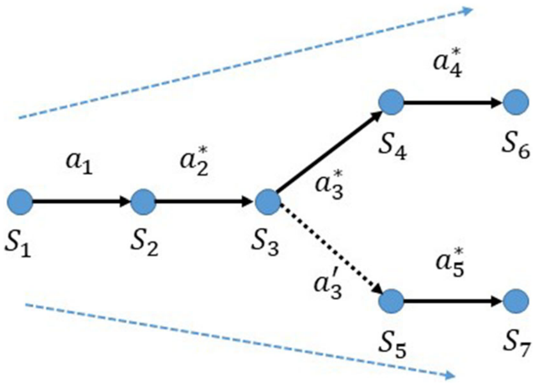


Fig. 1 The generation process of greedy action chains, where the solid lines with arrows form the greedy action chains

backpropagation of error updates through the tree structure and *Q*-value updates, data packets are sent using greedy policies, and reinforcement learning is performed during data packet sending.

5.1 Greedy action chains

When applying reinforcement learning algorithms to routing problems in wireless sensor networks, such as the Q-routing algorithm, QELAR algorithm, and RLBR algorithm, each step of error update can only affect one node. However, when introducing the reinforcement learning algorithm based on eligibility traces to communication in wireless sensors, nodes can reversely propagate the error of one learning step along the learning trajectory for multiple nodes in a detection cycle. Because the Sarsa (λ) method (Sutton and Barto 1998; Jiang et al. 2019) is an online learning algorithm, its action trajectory is discarded after learning, and its *Q*-value is instantly updated by the node's message, as well as the *Q*-value of the explored next-hop node is also used to update the *Q*-value of the current node. While Q (λ) has a definite learning trajectory through the reverse learning process of optimum *Q*-value routing, from the learning trajectory, the entire wireless sensor network produces many state-action chains, and when an exploring action is encountered, this state-action chain is disrupted.

As illustrated in Fig. 1, in the state S_1, the agent chooses the action a_1 into the state S_2, $Q(S_1, a_1)$ depending on $Q(S_2, a_2^*)$ where the greedy action is chosen in the state S_2, and the action with the highest value of Q is chosen in the optional states. The agent enters the state S_3 after performing the greedy action a_2^* in the state S_2. The update of $Q(S_2, a_2^*)$ is dependent on $Q(S_3, a_3^*)$, where a_3^* is the greedy action done in the state S_3, and the action with the highest Q value is chosen in the optional state. The agent then reaches the state S_5 after performing the exploration action $a_3^{'}$ in state S_3. The update of $Q(S_3, a_3^{'})$ is dependent on $Q(S_5, a_5^*)$, where a_5^* is the greedy action executed in the state S_5 and the action with the highest Q value is chosen in the optional state.

The agent's actual trajectory is as follows:

$$S_1 \rightarrow (a_1) \rightarrow S_2 \rightarrow (a_2^*) \rightarrow S_3 \rightarrow (a_3^{'}) \rightarrow S_5 \rightarrow (a_5^*) \rightarrow S_7$$

The backpropagation trajectory of the Q-value error is as follows:

$$Q(S_1, a_1) \leftarrow Q(S_2, a_2^*) \leftarrow Q(S_3, a_3^*) \leftarrow Q(S_4, a_4^*)$$

Only greedy actions directly impact Q-value updates; non-greedy actions' primary job is exploration, which does not affect the backpropagation of the current Q-value. A greedy action chain is a trajectory formed by an agent which has chosen greedy actions since its beginning. Our work aims to discover a greedy action chain and store it in network nodes, allowing error backpropagation over multiple steps to increase learning efficiency.

5.2 Reinforcement learning routing with tree-based eligibility traces (RLR-TET)

Wireless sensor nodes communicate with one another and plan their paths by exchanging control packets before transmitting data packets along the path. Previously, each node in a wireless sensor network sent probing information to surrounding nodes and updated the Q table entries with feedback response information. Because this probe packet contains no data elements, it is small in length and uses few network resources. This probe information must be sent on a regular schedule, and if the time interval is set to 10 s, such a period can only learn 1 step every 10 s, resulting in a learning efficiency that is too low and a convergence speed that is too slow to cope with the network's topological changes over time. If the Q table is not sufficiently learned, it might result in incorrect packet delivery. To address this issue, if each node's probing interval is too short, it will result in data package congestion and network energy imbalance.

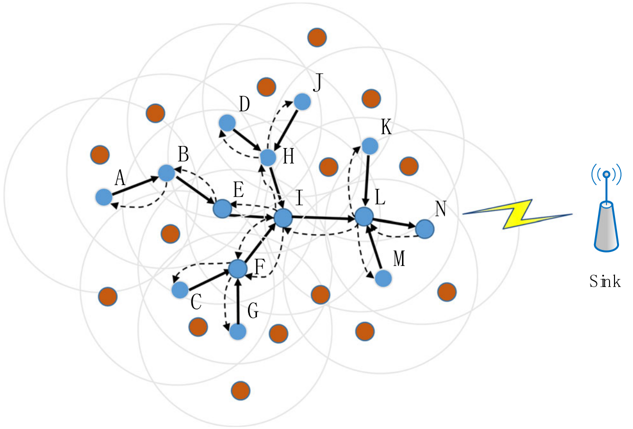


Fig. 2 A tree of greedy action chains in wireless sensor networks. The solid line indicates the direction of packet transmission, and the dashed line indicates the direction of error propagation. A single V-value update of a node can be backpropagated through the tree to multiple nodes

Using the greedy action chains presented in Sect. 5.1, we may reverse propagate the learning error of a single node to many nodes along the greedy action chain in a triggered manner by computing eligibility traces and updating Q-values during 10 s. This method can improve learning efficiency.

As illustrated in Fig. 2, we present a wireless routing protocol with error backpropagation using eligibility traces, where the greedy action chains form a tree with sink as the root. Each node stores the next-hop nodes within the communication range and their Q-values, from which the V-values may be derived.

For any node i, there exists the possibility that the node j's greedy action is to send data to node i. A set J is formed by all nodes j that meet the criterion. When node i receives a V-value update, all the Q-values to node i of nodes in the set J are updated, then V-values are computed. Subsequently, numerous node Q table entries are changed along the tree in reverse order.

The RLR-TET algorithm improves on the earlier "probe and reply" strategy, in which each node broadcasts control information to its neighbors during each control cycle, and then the neighboring nodes immediately learn and update the associated Q values upon receiving the control packets. Each node transmits a control packet message to its neighboring nodes during a control period T_w. The message includes the following information: message type, ID number, control packet ID number, remaining energy, the minimum number of hops to sink, V value, coordinate value, and next-hop optimum transmission node.

Setting node i's neighbors as neighbor (i). Node i delivers control packets to its neighbor nodes regularly, and for each node $j \in$ neighbor (i), the neighbor nodes of node i compute the Q (j, i) value using the information in the control packets given along.

Corresponding to the Markov decision process, the information of node j is considered as the agent's state. If the data packet is transferred from node j to node i, it is regarded as the $a_{j,i}$ action which yields the reward value, including the following five parts.

(1) The reward value r is acquired through the state transfer. $r = 100$ if the node transferred to is the sink, $r = -10$ if there is no neighbor node to transfer to, and $r = 0$ otherwise.

(2) The number of hops between node i and the sink: minHopCount.

(3) Node i's remaining energy: eRemain.

(4) Node j's transmission power: transmissionPower. It may be calculated using formula (8) based on the distance between nodes j and i.

(5) Node i’s received power: receivePower. It is obtained by a formula (9).

The data packet is sent to node i from node j, as specified above. Its iterative updating formula for Q value is as follows:

$$\mathrm{Q}\left(\mathrm{j}_{\mathrm{t}+1},\mathrm{i}_{\mathrm{t}+1}\right)=(1-\alpha)\mathrm{Q}(\mathrm{j}_{\mathrm{t}},\mathrm{i}_{\mathrm{t}})+\alpha[\mathrm{w}_1{*}\mathrm{r}+\mathrm{w}_2{*}\mathrm{minHopCount}+\mathrm{w}_3{*}\mathrm{eRemain}+\mathrm{w}_4{*}\mathrm{transmissionPower}+\mathrm{w}_5{*}\mathrm{receivePower}+\gamma V_{\mathrm{i}_{\mathrm{t}}}\Big] \tag{11}$$

Equation (11) is used to determine the $Q(j,i)$ values for all nodes from node $j \in$ neighbor (i) to node i, where w_1,w_2,w_3,w_4, and w_5 are learning parameters.

The V-value of a node is determined from the Q-value when it receives information from its neighbors. The following formula determines the V-value of any node i:

$$V_i = \max_{j\in\mathrm{neighbor(i)}} Q(i,j) \tag{12}$$

Based on the set of Q-values of node i, find and record the node with the highest Q-value among the neighboring nodes.

$$\mathrm{greedyNode} = \operatorname*{argmax}_{j\in\mathrm{neighbor(i)}} Q(i,j) \tag{13}$$

Because the sensor network is discrete, each node cannot acquire the minor hop count to the sink directly, and the minimum hop count to each node is frequently modified owing to network topology changes. Each node’s data structure contains a minHopCount item that records its minimal hop count to the sink. The sink’s minHopCount item is set to 0, while the rest of the nodes’ minHopCount items are initialized to infinity, which is set to 9999 for convenience of computation. In each cycle, including the sink, the node broadcasts a control packet to its neighboring nodes, and the neighboring nodes receive the control packet and calculate the minimum number of hops to the sink. After several cycles of propagation, each node acquires a more accurate minHopCount value. The sensor nodes in the actual environment broadcast information across the successor nodes and trigger the previous-hop node to undertake the update action.

The least number of hops to the sink is used in the reward value computation, which always induces the system to send data via the shortest path. In this paper, the parameter w_2 gradually decays with increasing time to equalize network power, shown in Eq. (14).

$$w_2(t) = w_2(t_0){*}e^{-t/\tau} \tag{14}$$

where $w_2(t_0)$ is the initial parameter and τ is the period constant. The smaller the value of τ, the quicker $w_2(t)$ decays, while the higher the value of τ, the slower $w_2(t)$ decays.

Its iteration mechanism is as follows: for each node $j \in$ neighbor (i), if the hop count $\mathrm{minHopCount}_i + 1$ from node i to the sink is less than the hop count $\mathrm{minHopCount}_i + 1$ from node j to the sink, then set $\mathrm{MinHopCount}_j = \mathrm{MinHopCount}_i + 1$.

Nodes in the network use the reinforcement learning process to compute Q-values from the current node to each neighboring node, and the greatest value from these Q-values is obtained as the V-value using Eqs. (12–13). The optNextNode field keeps track of the nearby node with the highest Q-value, and if many surrounding nodes have the same maximum Q-value, one is chosen randomly. In this manner, greedy action chains connect the whole network into a tree structure. Thus, through real-time learning, the challenge of optimizing routing rules for sensor networks is turned into developing and maintaining one or more efficient trees balanced in power consumption and load. The optimal routing strategy begins at each node and goes in the direction of the arrow in Fig. 2. For example, if a data packet is transferred from node A to the sink, the packet transmission path is A-B-E-I-L-N, and the packet is then sent from node N to the sink.

During the learning process, changes in node power and the minimum number of hops from a node to the sink might induce variations in the Q value. At this point, the tree structure is modified by simply changing the value of the optNextNode field. This process establishes each node’s Q-value and the computational model of tree-based error backpropagation. Based on the tree, each node must keep track of the set of its direct predecessor nodes, which is referred to as optLastNodes. In Fig. 2, for example, nodes H, E, and F all point to node I. Nodes H, E, and F are the node i’s immediate predecessors. Node I then logs the group of nodes H, E, and F.

If the V-value of a node changes during the reinforcement learning process, the resulting error value will update the Q-value of its predecessor nodes by using the eligibility traces in the opposite directions of the arrows in the trees and then continue to propagate the updates to the previous level predecessor nodes. The rule of error propagation states that whichever node refers to this node, this node propagates the error backward via the eligibility traces to whomever. The number of error propagation levels should usually not exceed 4 to 6 levels of the tree, depending on the size of the network, in order to keep the amount of information in the network low.

Algorithm 1 Backpropagation of errors via tree with eligibility traces

```
backUpdateQ(α, γ, λ, e, i, δ, backPropagationLevel)
if backPropagationLevel<=0 && i==NIL then
    exit();
end if
optLastNodes =i.optLastNodes;
for each optLastNode in optLastNodes
    Q(optLastNode,i)= Q(optLastNode,i) + αδe;
    e=γλe;
    backUpdateQ(α, γ, λ, e, optLastNode, δ, backPropagationLevel-1);
end for
```

Individual nodes in a dispersed environment do not need to know the global state of network nodes; instead, they must connect with surrounding nodes using the approach described in this paper to get the optimal data packet transmission path rapidly. The algorithm for learning individual node routes is depicted in Algorithm 1.

This work presents an approach for propagating the V-value errors of nodes through the tree utilizing eligibility traces in Algorithm 1, and the update is accomplished by reversing the greedy action chains in a one-to-many error propagation process. Previous approaches only conduct one step of learning and updating in one cycle. However, Algorithm 1 allows many nodes to learn and update in a triggered way in one cycle through a tree composed of greedy action chains.

In the backUpdateQ function, the following arguments are defined: α is the learning rate, γ is the discount factor, λ is the eligibility traces factor, e is the eligibility traces, δ is the error caused by the V value in node learning, and backPropagationLevel is the depth of propagation by error. The approach imposes several constraints on the propagation distance and the number of tree propagation layers. BackPropagationLevel is set to LIMITLEVEL levels in this work, and the value of LIMITLEVEL ranges from 3 to 5 layers. To explain the algorithm, Algorithm 1 uses a recursive method.

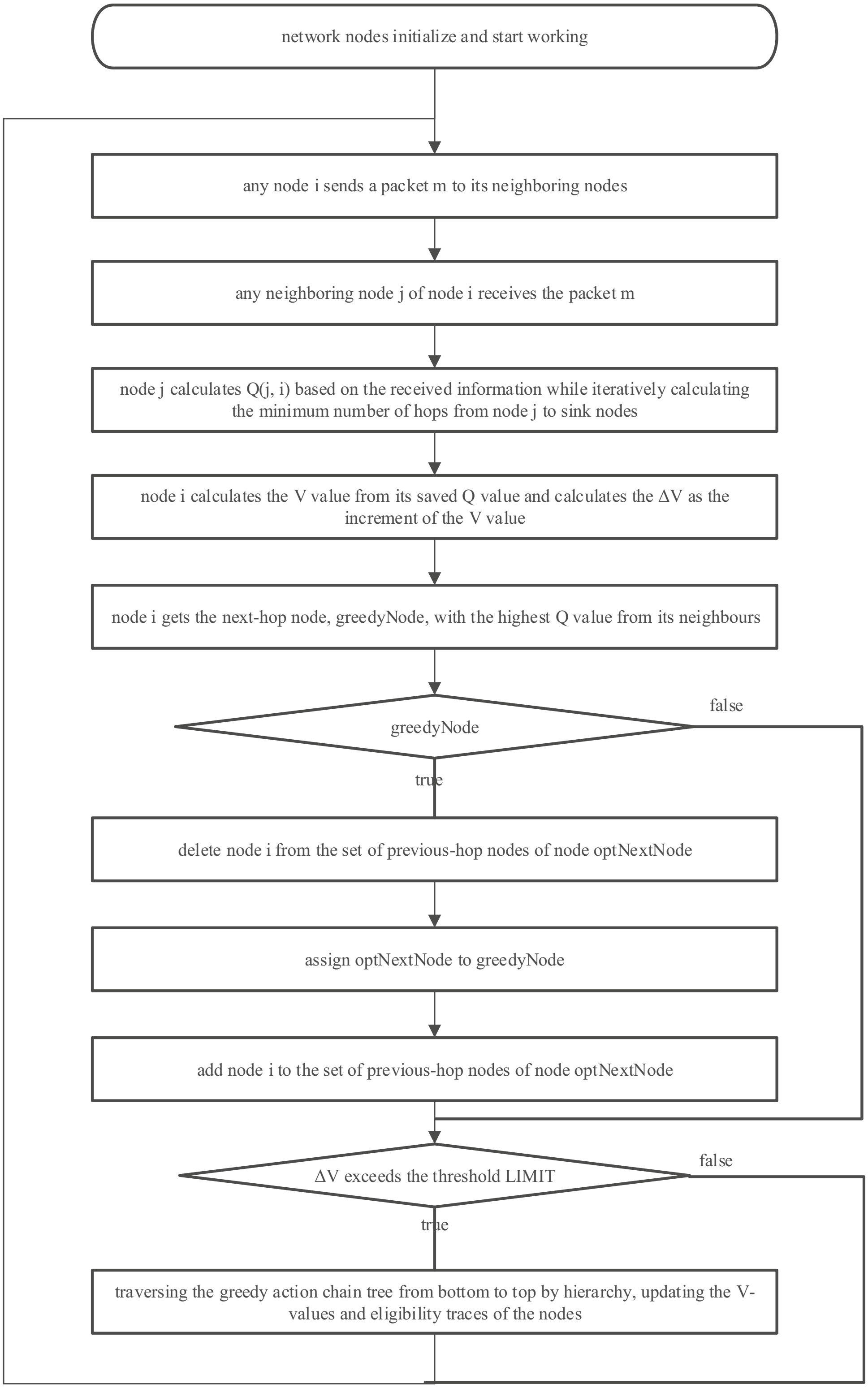
As the error is backpropagated hierarchically from the root of the tree, the Q value is updated in a proportionally decreasing manner due to the eligibility traces. As the error propagates back one layer, the trace e decreases to αγλe. By adjusting the parameters α, γ, λ, we can control the reinforcement learning process of the network. If the value of λ is larger, the error value of a node's V-value will affect a larger range of nodes; if the value of λ is smaller, the error value of a node's V-value will affect a smaller range of nodes.

Algorithm 2 Individual node route learning algorithm

```
Initialization the parameters: α, γ, λ, e =1, backPropagationLevel=LIMITLEVEL；
for each cycle do
    Node i sends a control packet to its neighbor nodes；
    for each node j∈neighbor(i)
        Calculate Q(j, i) for sending the message from node j to node i using equation (11);
        if j. minHopCount>i. minHopCount+1 then
            j.minHopCount=nextNode.minHopCount+1;
        end if
    end for
    for each node j∈neighbor(i) do
        Calculate $V_i$ value using equation (12)；
        $\Delta V_i$ = the new value of $V_i$ – the old value of $V_i$；
        Use equation (13) to calculate the greedyNode；
        if i.optNextNode≠ greedNode then
            Remove node i from the i.optNextNode. optLastNodes set;
            i.optNextNode= greedyNode;
            Add node i to the i.optNextNode. optLastNodes set;
        End if
        if $|\Delta V_i| > LIMIT$ then
            backUpdateQ(α, γ, λ, e, i , $\Delta V_i$ , backPropagationLevel);
        end if
    end for
end for
```

Each node is unaware of the network's global knowledge, and the network can only receive routing decisions through nodes talking with their neighboring nodes regularly and iterating step by step. Network choices are generated as a consequence of autonomous learning from each sensor node, with each node's algorithm being independent and identical. Algorithm 2 provides the sensor node routing algorithm. Each node broadcasts a control packet to its neighbors regularly, and when one of them gets it, the neighbor node changes the *Q* value and the value of minHopCount, which is set to one of the reward values. The *Q*-value of nodes to their neighbors is then calculated, and the greatest of them is discovered as the *V*-value, which contributes to greedy action chains. If the node i's *V* value changes and ΔV_i exceeds the LIMIT threshold, Algorithm 1 is invoked to start the learning process. The flowchart of the RLR-TET algorithm is shown in Fig. 3.

Fig. 3 Flowchart of the RLR-TET algorithm


network nodes initialize and start working
any node i sends a packet m to its neighboring nodes
any neighboring node j of node i receives the packet m
node j calculates Q(j, i) based on the received information while iteratively calculating the minimum number of hops from node j to sink nodes
node i calculates the V value from its saved Q value and calculates the ΔV as the increment of the V value
node i gets the next-hop node, greedyNode, with the highest Q value from its neighbours
greedyNode
false
true
delete node i from the set of previous-hop nodes of node optNextNode
assign optNextNode to greedyNode
add node i to the set of previous-hop nodes of node optNextNode
ΔV exceeds the threshold LIMIT
false
true
traversing the greedy action chain tree from bottom to top by hierarchy, updating the V-values and eligibility traces of the nodes

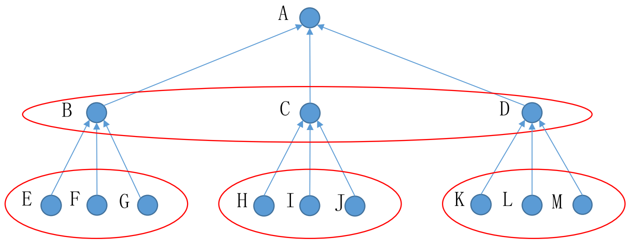


Fig. 4 A clearer greedy action chain tree structure. The first level contains **A**; the second level contains **B**, **C**, and **D**; the third level contains **E**, **F**, **G**, **H**, **I**, **J**, **K**, **L**, and **M**

5.3 Performance analysis of the RLR-TET algorithm

In a tree of greedy action chains, the root node is *A*, which forms the first level of the tree. There are m nodes in the neighbors of node *A* that take node *A* as the optimal next-hop node, and the set S of these m nodes forms the second level of the tree. Based on all the nodes in the second level, the nodes in the third level of the tree are obtained. In the same way we can get the deeper nodes.

Assume that each node corresponds to a set with a number of nodes *m*, while the tree has L levels. Using the RLR-TET algorithm, during a control packet transmission cycle, if the *V* value of the root node *A* changes, the error is hierarchically backpropagated, modifying the *Q* values of the sensors. The number of sensors updated in this process is as follows:

$$m + m^2 + m^3 + \cdots + m^P = \frac{1 - m^P}{1 - m} \tag{15}$$

For the sake of discussion, we set the number of nodes in the set corresponding to each node to *m*. In fact, this number is variable. We assume that *m* = 3 and *P* = 3. This is shown in Fig. 4. By calculation, during a control cycle, the number of nodes that can update the *Q* values is 13.

Algorithm 3 Transmitting data packets for a single sensor node

```
Initialization the parameters: α, γ, λ, e =1, backPropagationLevel=LIMITLEVEL；
if Sensor node i generates a data packet or receives a data packet from a neighboring node then
    optNextNode=i.optNextNode;
    Send the data packet to the optNextNode node；
    wait until Sensor node i receives the reply packet；
    Use formula (11) to update Q(i,optNextNode);
    $\Delta V_i =$ the new value of Q(i,optNextNode)−the old value of Q(i,optNextNode);
    if $|\Delta V_i| > LIMIT$ then
        backUpdateQ(α, γ, λ, e, i , $\Delta V_i$ , backPropagationLevel);
    end if
end if
```

5.4 Reinforcement learning is used in the data packet transmission process

Two types of packets are transmitted between nodes in reinforcement learning-based wireless sensor node routing. The first type is the control packet, which, as mentioned in the previous section, has the primary function of constructing and updating the Q table to actualize decision and control of data packet transmission, and this control packet will not be forwarded and delivered via nodes. The second type of packet is a data packet delivered between nodes. Unlike the control packet, the data packet is processed completely under the control of the Q table for routing transmission without the exploration process, and its data size is substantially bigger than that of the control packet. Meanwhile, during data packet transmission, if the packet is sent from the current node to the next-hop node and the V value of this node is updated, and the $|\Delta V|$ is greater than the specified threshold, it also triggers the reinforcement learning process, which propagates the error information backward through the tree and affects multiple nodes to complete the Q value update. Control packets are shorter and cause less energy loss to network nodes; data packets are the principal component of network information transmission and have a big data size, which is the leading cause of energy loss and rough energy usage in network nodes. Algorithm 3 demonstrates the data packet transmission algorithm for a single sensor node.

In Algorithm 3, a node transmits a packet, where Q(i,optNextNode) in the greedy action chains tree is the V value of node i. During the learning process, the node's Q-values change, and the node's V-value is updated using Algorithm 2, along with the tree structure. During the transmission process of delivering a data packet to the sink, the data packet is regarded as an agent whose trajectory is the data packet's transmission path, and the agent is also continually learning from the reply packet. Because of its vast number of packets, it is the most significant contribution to node energy loss and network congestion in data transmission. In this paper, we define the maximum number of hops of packet delivery, after which the network nodes immediately discard the packet. It can be noticed that the accuracy of decision knowledge is highly significant in the reinforcement learning process, as well as the efficiency should be high enough to achieve the network optimization strategy, which is the principal purpose of this paper's research.

In wireless sensor networks, a tree composed of greedy action chains is built by reinforcement learning algorithms and dynamically modified during the learning process. It serves two essential functions: first, each node's packet directly obtains the optimal next-hop node via its optNextNode term, which points to the node leading to the root of the tree (sink); Second, the direction of the learning error update is the opposite direction of the optNextNode term of each node, which is the set of the optLastNode term.

6 Simulation results and discussion

We created a simulated environment for wireless sensor networks and tested the algorithm suggested in this paper. Although the method validation is done in a simulated environment, it may also be used in a real-world scenario. This simulation accurately replicates the physical parameters in the actual scenario. However, it is not a problem to have some deviations in the real scenario of wireless sensor networks, as the adaptive algorithm using reinforcement learning will reasonably adjust the parameters to satisfy such changes. Table 3 shows the physical parameters we used for the experimental setting.

The RLR-TET method is validated and compared with other algorithms in this paper. In a 1000m × 1000m region, 98 sensor nodes are randomly and uniformly arranged, and their initial power is all 10 J; in the meantime, there are two sinks in the area with unconstrained energy. To validate the algorithm in this paper, we set the sensor nodes to be stationary during operation. During the continuous learning process, each node does not know global information and only interacts on a limited scale with neighboring nodes to generate optimum routes in the wireless sensor network. The links between nodes are symmetric, and there is no case in which node A may receive information from node B and node B cannot receive information from node A. This experiment has two scenarios in which the data packets are dropped during transmission. The first scenario is that when the data packet is transmitted with many hops greater than the allowed number of hops, it will be deleted by the node because the packet cannot find the path. The second scenario is that adjacent nodes die due to power consumption during data packet transmission, and no node can approach the sinks via data packet communication. The first issue can be avoided by immediately determining an optimal path with reinforcement learning algorithms. The power of network nodes should be adjusted as much as possible so that individual nodes do not die early, avoiding the second condition. The maximum radius of a node's one-hop communication is set at 100 m, as shown in Table 2

In the experiment, the node communication strategy must be continually adjusted and optimized based on the node's transmission power, receiving power, remaining power, number of hops, and distance to the sink. Each node sends a control packet to its neighboring nodes every 10 s to make communication decisions, which contain V value, power, and the minimum number of hops known to the sinks. The neighboring nodes calculate the Q-values after receiving the control packet. Because the control package is tiny, it consumes relatively little energy. Every node learns every 10 s, and the network is continually updated with the tree built by greedy action chains. The learning goal is for each node to discover the best route to the sinks. To assess the algorithm's performance, nodes 1, 3, 7, 9, 23, 26, 27, 46, 67, 76, 91, 95, 98, and 100 are used as sensing nodes to generate sensing data for delivery to sinks, while the other nodes are exclusively responsible for routing

Table 3 Parameters setting

Network parameters	Values
Network area	1000 m × 1000 m
Number of nodes	98
Number of sinks	2
Initial energy	10 Joules
E_{elec}	50nJ/bit
ε_{fs}	10pJ/bit/m^2
ε_{amp}	0.0013pJ/bit/m^4
Control package size	48bits
Packet size	512Bytes
Node transmission radius	100 m
Distance threshold d_0	87 m
Control packet sending cycle	10 s
LIMITLEVEL	4
time-to-live (TTL)	8
The cycle of control packets sent by the node	10 s
The interval between packets generated by the source node	10 s
Learning rate α	0.1
Discount factor γ	0.85
Eligibility traces factor λ	0.6

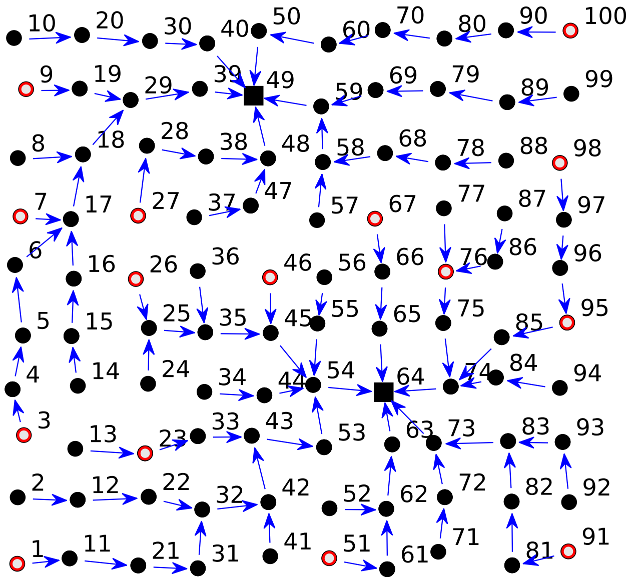


Fig. 5 Network path diagram after 7 h. In learning, each sensor that can only sense local area nodes is able to route the packet through the optimal path to the next hop node and eventually to the sinks. Shorter paths and power balanced routing are achieved

forwarding. Nodes 49 and 64 are sinks with adequate power and good communication performance by default. Set the sensing node to generate a data packet every 10 s, with a packet size of 512 Bytes, and the sensor node forwards data at 0.1-s intervals until it reaches the sinks. The transmission process of this packet is terminated if the number of transmission hops is surpassed or if the relay node lacks sufficient power to advance.

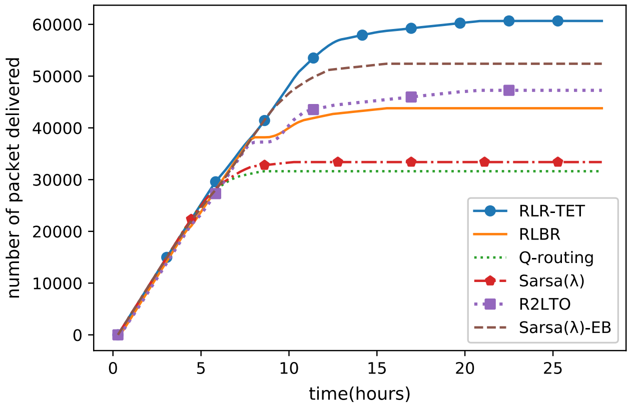


Fig. 6 The number of data packets delivered to sinks varies with time. Because the RLR-TET algorithm enables the network to achieve Q-value updates quickly, nodes can quickly adopt optimized data transmission strategies and the network maintains energy balance for a longer period of time. As a result, RLR-TET is able to transmit more packets

The RLR-TET algorithm, whose aim of path planning considers not only the short transmission path of data packets but also the power consumption, assures the power balance of the whole network while choosing the next-hop node among neighboring nodes to extend the network's working life. The network path diagram after 7 h is shown in Fig. 5, and the blue line segments with arrows depict node routing. Starting with the sink node, it is evident that learninSg causes the network to construct a tree comprising greedy action chains, by which the error propagates backward. The RLR-TET algorithm is compared to the Q-routing algorithm, the RLBR algorithm, the R2LTO algorithm, and the Sarsa(λ) algorithm in this research to validate it. The comparison metrics include the number of data packets transferred to sinks with time variation, the number of alive nodes with time variation, and the standard deviation of the network nodes' remaining energy with time variation.

6.1 The number of data packets delivered to sinks varies with time

The various algorithms are tested until the network cannot finish any data packet transmissions. As illustrated in Fig. 6, the number of packets delivered by the various algorithms was nearly identical from the moment the network went live until 5.2 h later. After 5.2 h, the number of packets delivered by the Q-routing and Sarsa(λ) algorithms begin to decline because these two algorithms do not consider energy balancing and only learn by the shortest path as the goal. These two methods do not uniformly schedule the energy in the path planning process, causing the nodes on the shortest path to run out of power prematurely, prompting the network to fail and the sensor nodes to be unable to discover an appropriate route to reach the target sinks. The Sarsa(λ) method outperforms the Q-routing algorithm for the following reasons: during the learning phase, Sarsa(λ) is more efficient than Q-routing and can identify the shortest path faster. The same packet is transmitted to the sink, where fewer relay nodes are involved, conserving network node energy and increasing network lifetime, allowing the network to deliver more packets to sinks to some extent. However, because both methods only evaluate the shortest route transmission, the network cannot work efficiently when more than 30,000 packets have been successfully delivered.

The RLBR and R2LTO algorithms consider the nodes' remaining energy, the distance between them, and the number of hops from the nodes to the sink. These two approaches learn to adapt to changing conditions, resulting in energy-balanced routing, the prevention of premature node power depletion, and the extension of network lifetime, which leads to increased packet transmission quantity.

The residual energy of nodes and the energy consumption of communication between nodes are not considered in the Sarsa(λ) method learning process. The remaining energy of the node and the energy consumption of node communication are introduced to the learning process of the Sarsa(λ) method in this paper, which is named Sarsa(λ)-EB (Sarsa(λ) with energy balance). Sarsa(λ)-EB method significantly increases network lifetime, increasing the total number of successful packet transfers. The number of successfully transferred data packets has been raised experimentally from 33,395 to 52,397. Considering the nodes' residual and communication energy, the RLR-TET algorithm increases the network's learning efficiency even more. The approach propagates the V-value error in node learning through greedy action chains to multi-layer nodes in a tree as rapidly as possible to develop an effective network transmission policy for routing. Several other energy-based algorithms have worse learning efficiency than the RLR-TET method. The total number of transmitted packets by the RLR-TET algorithm reaches 60,665, significantly improving over the Sarsa(λ)-EB algorithm.

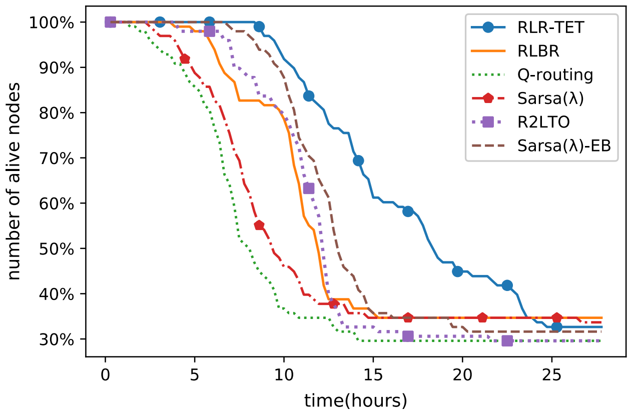


Fig. 7 The number of alive nodes with the time variation. As the RLR-TET algorithm considers the remaining energy of each node, the energy consumption is more balanced, so more nodes are kept alive during the operation of the network

6.2 The number of alive nodes with the time variation

Figure 7 shows the number of alive nodes with the time variation in the network. After running the Q-routing and Sarsa(λ) algorithms in the network for 1.2 and 2.4 h, respectively, the first node that runs out of power appears in the network. Both methods always attempt to send packets through the shortest path, and because the reinforcement learning process is adaptive, both algorithms can adjust to find another shortest path when a node runs out of energy and drops out of the network. When combined with the results in Fig. 6, the loss of a small number of nodes has little effect on packet routing in the network. The number of surviving nodes then begins to drop, and data packets can no longer reach the sinks after around 5.1 h. At this point, data packets are still attempting to reach sinks, draining the energy of the nodes until a significant number of isolated nodes form and the network's lifetime ends.

Using the RLBR and R2LTO algorithms, the first node that runs out of energy appears around 3.5 h, and the overall quantity of data transmission begins to inflect about 5.3 h, after which the performance rapidly falls.

Because of the enhanced learning efficiency, the Sarsa(λ)-EB algorithm demonstrated the first node that ran out of energy after just 6.5 h, following which the performance improved significantly due to the achievement of balanced energy scheduling.

The RLR-TET algorithm has a high learning efficiency by constructing an error backpropagation tree comprised of greedy action chains during the network learning process and dynamically modifying the tree as the nodes' energy changes. With this approach, the first node that runs out of energy appears after 8.3 h, and the network's lifetime is considerably increased.

6.3 The standard deviation of the network nodes' remaining energy with its time variation

The standard deviation of the remaining energy in the nodes accurately measures the network's energy balance. Because the energy of sinks is not constrained, it is not utilized to determine the standard deviation. Furthermore, the 15 sensor nodes generate detection data and transfer it every 10 s, consuming energy at regular intervals. To more precisely assess the influence of various reinforcement learning algorithms on network performance, these nodes' electric power is also not considered when calculating network nodes' standard deviation. As a result, the standard deviation is calculated using 83 nodes. As shown in Fig. 8, the Q-routing algorithm and the Sarsa(λ) algorithm, which do not take into account the electrical energy metrics, show that the standard deviation of network energy is continuously increasing from the start of network operators to 350 min, indicating that the node energy imbalance is becoming more apparent. The standard deviation of the network learned using the RLBR algorithm, and the R2LTO algorithm is lower than that of the Q-routing algorithm and the Sarsa(λ) algorithm, indicating that the system tends to adjust to the standard deviation automatically. The standard deviation of the Sarsa(λ)-EB algorithm is generally less than 1.5, which shows that the error multi-step backpropagation reinforcement learning algorithm has a significant effect on balancing the node energy. The RLR-TET algorithm's standard deviation has always been less than 1.0. The standard deviation has been effectively controlled in Fig. 8, revealing that the method in this paper achieves a better network energy balance, which is considerably better than the other five methods.

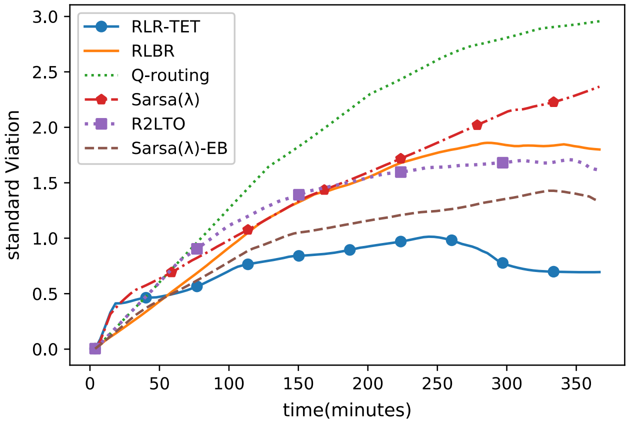


Fig. 8 Standard deviation of the network nodes' remaining energy with its time variation. The standard deviation is an ideal indicator of network balance. The figure shows that the RLR-TET algorithm is able to keep the standard deviation to a small value

6.4 Discussion of experimental results

From the analysis of the experimental results, the RLR-TET algorithm can achieve fast iteration of Q values and has excellent learning performance. By using the RLR-TET algorithm, the nodes have a better data delivery strategy. It has a shorter path while maintaining the energy balance of the network nodes. This is because our method propagates the errors generated by nodes in the learning process backwards to a larger number of nodes with maximum efficiency, allowing the entire network of nodes to quickly sense and update their Q values and obtain better learning results. Figure 8 shows by standard deviation that the RLR-TET algorithm allows the network nodes to be energy balanced. This reduces the number of sensors that run out of energy prematurely, which also maximizes the connectivity of the network and prevents it from failing prematurely. A robust network will transmit more packets, as shown in Fig. 6.

In this algorithm, the sensor node has to send an additional control packet, which results in additional data consumption. However, this control packet consumes very little power compared to the data packet. The lifetime of the sensor nodes is also significantly increased due to the fast generation of optimal decisions. If a packet cannot find a route to the sink and is forced to be dropped, this not only causes the network to lose data, but also results in higher power consumption of the sensors.

7 Conclusion

The RLR-TET algorithm improves the energy balancing problem in wireless sensor network routing, in which sensor nodes broadcast control packets to neighboring nodes at regular intervals, and neighboring nodes update their Q-values and then V-values after receiving the control packets. The network gradually builds greedy action chains during the learning process, which dynamically form a tree structure in the network and propagate the error backward along the tree when the node's V value is updated. Such an update of a node's Q or V value causes an update of the Q values of multiple sub-nodes in the network, resulting in a rapid update of the network Q table to quickly adapt to the changing network environment and provide a more efficient decision-making action for data transmission. The RLR-TET algorithm updates the network using a periodic and triggered approach that does not consume too much of the network's adequate time. The transmission of network information includes both the transmission of control packets and the transmission of data packets, and the Q value of nodes is updated in both modes. There is no exploration during data packet transmission, and they are sent from the child node to the parent node entirely according to the tree formed by greedy action chains, avoiding packet loss during the exploration process. The RLR-TET algorithm improves the reinforcement learning algorithm and achieves better results by adapting to the specific situation of wireless sensor networks. The comparison algorithms in this paper are the Q-routing algorithm, the Sarsa(λ) algorithm, the RLBR algorithm, the R2LTO algorithm, and the Sarsa(λ) -EB algorithm, all of which are typical routing methods for wireless sensor networks that use reinforcement learning. The methods described in this paper compare metrics such as the total number of packets transmitted to sinks, the number of surviving nodes, and the standard deviation of node energy. With the methods described above, it is clear that the RLR-TET algorithm outperforms the others. In order to validate the effectiveness of the algorithm and to facilitate comparison with other methods, the scenarios of node movement, node data buffer, channel selection, channel interference, and communication security are not considered in this paper; however, future work will extend our research to these scenarios and explore the corresponding algorithms. Considering these QoS metrics would make the study more complex, but more realistic. However, we can expect that the RLR-TET algorithm will still produce good results, and this is worthy of further research. The sensors discussed in this article are not rechargeable and their energy will continue to diminish. Therefore, the aim of our research is to save battery power and balance network power. However, if the battery of the sensor node is rechargeable (Al-Janabi et al. 2020a, 2020b), the research idea of this paper would not be appropriate and its objective or reward function would have to be reset.

Funding This work was partly supported by the National Natural Science Foundation of China (Grant no. 60971088) and partly by the Natural Science Foundation of Shandong Province (Grant no. ZR2021MF013).

Data availability Data will be made available on request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

Al-Janabi S, Mohammad M, Al-Sultan A (2020a) A new method for prediction of air pollution based on intelligent computation. Soft Comput 24:661–680. https://doi.org/10.1007/s00500-019-04495-1

Al-Janabi S, Alkaim AF, Adel Z (2020b) An innovative synthesis of deep learning techniques (DCapsNet & DCOM) for generation electrical renewable energy from wind energy. Soft Comput 24:10943–10962. https://doi.org/10.1007/s00500-020-04905-9

Al-Janabi S, Alkaim A, Al-Janabi E, Aljeboree A, Mustafa M (2021) Intelligent forecaster of concentrations (PM2.5, PM10, NO2, CO, O3, SO2) caused air pollution (IFCsAP). Neural Comput Appl 33:14199–14229. https://doi.org/10.1007/s00521-021-06067-7

Basagni S, Valerio VD, Gjanci P, Petrioli C (2019) MARLIN-Q: multi-modal communications for reliable and low-latency underwater data delivery. Ad Hoc Netw 82:134–145. https://doi.org/10.1016/j.adhoc.2018.08.003

Bouzid SE, Serrestou Y, Raoof K, Omri MN (2020) Efficient routing protocol for wireless sensor network based on reinforcement learning. Int Conf Adv Technol Signal Image Process. https://doi.org/10.1109/ATSIP49331.2020.9231883

Boyan JA, Littman ML (1993) Packet routing in dynamically changing networks: a reinforcement learning approach, In: Proceedings of the 6th international conference on neural information processing systems, Morgan Kaufmann Publishers Inc., Denver, Colorado, pp 671–678

Chettibi S, Chikhi S (2016) Dynamic fuzzy logic and reinforcement learning for adaptive energy efficient routing in mobile ad-hoc networks. Appl Soft Comput 38:321–328. https://doi.org/10.1016/j.asoc.2015.09.003

Cho J, Lee H (2020) Dynamic topology model of Q-learning LEACH using disposable sensors in autonomous things environment. Appl Sci 10:9037. https://doi.org/10.3390/app10249037

Das SR, Beldingroyer EM, Perkins CE (2003) Ad hoc on-demand distance vector (AODV) Routing

Ding R, Xu Y, Gao F, Shen XS, Wu W (2019) Deep reinforcement learning for router selection in network with heavy traffic. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2904539

Frikha MS, Gammar SM, Lahmadi A, Andrey L (2021) Reinforcement and deep reinforcement learning for wireless Internet of Things: a survey. Comput Commun 178:98–113. https://doi.org/10.1016/j.comcom.2021.07.014

Gazi F, Ahmed N, Misra S, Wei W (2021) Reinforcement learning-based MAC protocol for underwater multimedia sensor networks. ACM Trans Sen Netw. https://doi.org/10.1145/3484201

Guo W, Zhang W (2014) A survey on intelligent routing protocols in wireless sensor networks. J Netw Comput Appl 38:185–201. https://doi.org/10.1016/j.jnca.2013.04.001

Guo W, Yan C, Lu T (2019) Optimizing the lifetime of wireless sensor networks via reinforcement-learning-based routing. Int J Distrib Sens Netw 15:155014771983354. https://doi.org/10.1177/1550147719833541

Hu T, Fei Y (2010) QELAR: a machine-learning-based adaptive routing protocol for energy-efficient and lifetime-extended underwater sensor networks. IEEE Trans Mob Comput 9:796–809. https://doi.org/10.1109/PCCC.2008.4745119

Jiang H, Gui R, Chen Z, Wu L, Zhou J (2019) An improved sarsa(λ) reinforcement learning algorithm for wireless communication systems. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2935255

Jin W, Gu R, Ji Y (2019) Reward function learning for Q-learning-based geographic routing protocol. IEEE Commun Lett. https://doi.org/10.1109/LCOMM.2017.2656879

Johnson DB, Maltz DA (1996) Dynamic source routing in Ad Hoc wireless networks

Jung W, Yim J, Ko Y (2017) QGeo: Q-learning-based geographic Ad hoc routing protocol for unmanned robotic networks. IEEE Commun Lett 21:2258–2261. https://doi.org/10.1109/LCOMM.2017.2656879

Kadhuim ZA, Al-Janabi S (2023) Codon-mRNA prediction using deep optimal neurocomputing technique (DLSTM-DSN-WOA) and multivariate analysis. Results Eng 17:100847. https://doi.org/10.1016/j.rineng.2022.100847

Karp B, Kung H (2000) GPSR: greedy perimeter stateless routing for wireless networks, In: Proceedings of the annual international conference on mobile computing and networking, MOBICOM, https://doi.org/10.1145/345910.345953

Kaur G, Chanak P, Bhattacharya M (2021) Energy-efficient intelligent routing scheme for IoT-enabled WSNs. IEEE Internet Things J 8:11440–11449. https://doi.org/10.1109/JIOT.2021.3051768

Kwon M, Lee J, Park H (2020) Intelligent IoT connectivity: deep reinforcement learning approach. IEEE Sens J 20:2782–2791. https://doi.org/10.1109/JSEN.2019.2949997

Li X, Hu X, Zhang R, Yang L (2020) Routing protocol design for underwater optical wireless sensor networks: a multiagent reinforcement learning approach. IEEE Internet Things J 7:9805–9818. https://doi.org/10.1109/jiot.2020.2989924

Li R, Li F, Li X, Wang Y (2014) QGrid: Q-learning based routing protocol for vehicular ad hoc networks, In: 2014 IEEE 33rd international performance computing and communications conference (IPCCC), 2014, pp 1–8 https://doi.org/10.1109/PCCC.2014.7017079

Lin D, Wang Q, Min W, Xu J, Zhang Z (2020) A survey on energy-efficient strategies in static wireless sensor networks. ACM Trans Sen Netw. https://doi.org/10.1145/3414315

Maleki M, Hakami V, Dehghan M (2017) A model-based reinforcement learning algorithm for routing in energy harvesting mobile Ad-hoc networks. Wireless Pers Commun. https://doi.org/10.1007/s11277-017-3987-8

Mammeri Z (2019) Reinforcement learning based routing in networks: review and classification of approaches. IEEE Access 7:55916–55950. https://doi.org/10.1109/ACCESS.2019.2913776

Mohammed GS, Al-Janabi S (2022) An innovative synthesis of optmization techniques (FDIRE-GSK) for generation electrical renewable energy from natural resources. Results Eng 16:100637. https://doi.org/10.1016/j.rineng.2022.100637

Oddi G, Pietrabissa A, Liberati F (2014) Energy balancing in multi-hop wireless sensor networks: an approach based on reinforcement learning. Adapt Hardw Syst. https://doi.org/10.1109/AHS.2014.6880186

Renold AP, Chandrakala S (2017) MRL-SCSO: multi-agent reinforcement learning-based self-configuration and self-optimization protocol for unattended wireless sensor networks. Wireless Pers Commun 96:5061–5079. https://doi.org/10.1007/s11277-016-3729-3

Saleh A, Ali B, Rasid M, Ismail A (2015) A self-optimizing scheme for energy balanced routing in wireless sensor networks using sensorAnt. Sensors. https://doi.org/10.3390/s120811307

Serhani A, Naja N, Jamali A (2020) AQ-Routing: mobility-, stability-aware adaptive routing protocol for data routing in MANET–IoT systems. Clust Comput 23:13–27. https://doi.org/10.1007/s10586-019-02937-x

Srivastava A, Mishra PK (2022) Multi-attributes based energy efficient clustering for enhancing network lifetime in WSN's. Peer-to-Peer Netw Appl 15:2670–2693. https://doi.org/10.1007/s12083-022-01357-w

Sun Y, Peng M, Zhou Y, Huang Y, Mao S (2019) Application of machine learning in wireless networks: key techniques and open issues. IEEE Commun Surv Tutor. https://doi.org/10.1109/COMST.2019.2924243

Sutton R, Barto A (1998) Reinforcement learning: an introduction

Wang P, Wang T (2006) Adaptive routing for sensor networks using reinforcement learning. IEEE Int Conf Comput Inf Technol. https://doi.org/10.1109/CIT.2006.34

Wang D, Liu J, Yao D (2020) An energy-efficient distributed adaptive cooperative routing based on reinforcement learning in wireless multimedia sensor networks. Comput Netw 178:107313. https://doi.org/10.1016/j.comnet.2020.107313

Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8:279–292. https://doi.org/10.1007/BF00992698

Yang H (2020) Machine learning techniques and a case study for intelligent wireless networks. IEEE Network. https://doi.org/10.1109/MNET.001.1900351

Yau K-LA, Goh HG, Chieng D, Kwong KH (2015) Application of reinforcement learning to wireless sensor networks: models and algorithms. Computing 97:1045–1075. https://doi.org/10.1007/s00607-014-0438-1

Zaraket E, Murad NM, Yazdani SS, Rajaoarisoa L, Ravelo B (2021) An overview on low energy wake-up radio technology: active and passive circuits associated with MAC and routing protocols. J Netw Comput Appl. https://doi.org/10.1016/j.jnca.2021.103140

Zhang M, Wang X, Liu T, Zhu J, Wu Q (2020) AFSndn: a novel adaptive forwarding strategy in named data networking based on Q-learning. Peer-to-Peer Netw Appl 13:1176–1184. https://doi.org/10.1007/s12083-019-00845-w

Zheng Z, Sangaiah AK, Wang T (2018) Adaptive communication protocols in flying Ad hoc network. IEEE Commun Mag 56:136–142. https://doi.org/10.1109/MCOM.2017.1700323

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.