technologies

MDPI

Article

A Reinforcement Learning-Based Dynamic Clustering of Sleep Scheduling Algorithm (RLDCSSA-CDG) for Compressive Data Gathering in Wireless Sensor Networks

Alaa N. El-Shenhabi [1,*], Ehab H. Abdelhay [1,2], Mohamed A. Mohamed [1] and Ibrahim F. Moawad [3,4]

[1] Department of Electronics and Communication Engineering, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt; ehababdelhay@mans.edu.eg (E.H.A.); mazim12@mans.edu.eg (M.A.M.)
[2] Faculty of Engineering, Mansoura National University, North Delta 35516, Egypt
[3] Department of Artificial Intelligence, College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia; ibrahim_moawad@cis.asu.edu.eg
[4] Faculty of Computer and Information Sciences, Ain Shams University, Cairo 11566, Egypt
* Correspondence: alaa_nabil@mans.edu.eg

Abstract: Energy plays a major role in wireless sensor networks (WSNs), and measurements demonstrate that transmission consumes more energy than processing. Hence, organizing the transmission process and managing energy usage throughout the network are the main goals for maximizing the network's lifetime. This paper proposes an algorithm called RLDCSSA-CDG, which is processed through the 3F phases: foundation, formation, and forwarding phases. Firstly, the network architecture is founded, and the cluster heads (CHs) are determined in the foundation phase. Secondly, sensor nodes are dynamically gathered into clusters for better communication in the formation phase. Finally, the transmitting process will be adequately organized based on an adaptive wake-up/sleep scheduling algorithm to transmit the data at the "right time" in the forwarding phase. The MATLAB platform was utilized to conduct simulation studies to validate the proposed RLDCSSA-CDG's effectiveness. Compared to a very recent work called RLSSA and RLDCA for CDG, the proposed RLDCSSA-CDG reduces total data transmissions by 22.7% and 63.3% and energy consumption by 8.93% and 38.8%, respectively. It also achieves the lowest latency compared to the two contrastive algorithms. Furthermore, the proposed algorithm increases the whole network lifetime by 77.3% and promotes data recovery accuracy by 91.1% relative to the compared algorithms.

Keywords: efficient energy; sparse-CDG; sleep scheduling; clustering; reinforcement learning; duty cycling; wireless sensor networks; energy harvesting

Received: 28 November 2024
Revised: 22 December 2024
Accepted: 26 December 2024
Published: 8 January 2025

Citation: El-Shenhabi, A.N.; Abdelhay, E.H.; Mohamed, M.A.; Moawad, I.F. A Reinforcement Learning-Based Dynamic Clustering of Sleep Scheduling Algorithm (RLDCSSA-CDG) for Compressive Data Gathering in Wireless Sensor Networks. *Technologies* **2025**, *13*, 25. https://doi.org/10.3390/technologies13010025

1. Introduction

Wireless sensor networks (WSNs) are systems made up of numerous small, distributed, and low-energy sensor nodes (SNs). These nodes are typically spread across a designated area to gather data from the target as shown in Figure 1, and then transmit them to the sink node. The functionality of a sensor node is typically divided into four basic units: processing, sensing, energy units, and transceiver. The processing unit is designed for managing operations and processing data collected by the sensing unit. The sensing unit observes and measures specific phenomena, such as temperature, humidity, motion, or other environmental variables. It captures the data that are crucial for the intended application. The energy unit provides the necessary power supply to the sensor node. The

transceiver unit enables SNs to communicate with other sensor nodes within the network, facilitating data exchange and coordination.

Unfortunately, most of the energy consumption in nodes occurs during the data transmission process. Hence, with the limited energy capabilities of sensor nodes, the efficient use of energy is a vital component of the WSN architecture, as shown in Figure 2. WSNs have expanded their applications to include surveillance, intruder detection, remote environmental monitoring, healthcare, industrial processes, space exploration, and agriculture. Today, WSNs function in dense environments and complex real-time applications. However, they face challenges such as limited energy, short lifespans, and poor channel bandwidth [1].

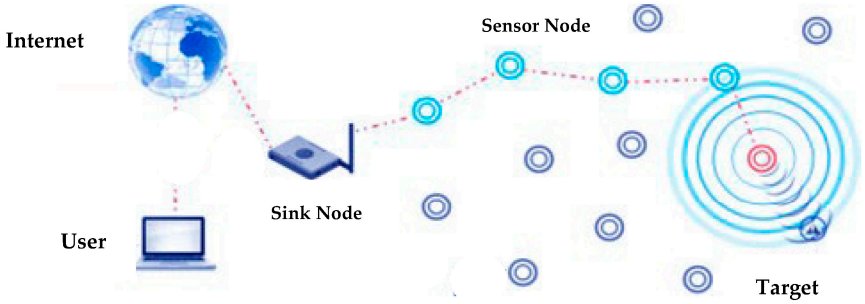

Figure 1. A typical of wireless sensor network architecture [2].

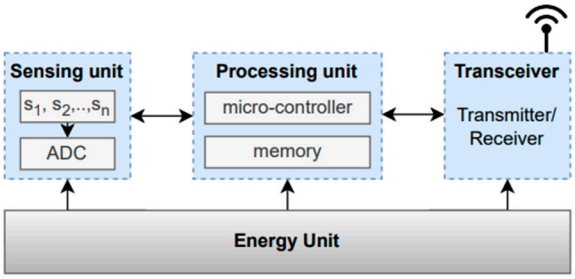

Figure 2. Sensor node architecture [3].

Fortunately, data compression innovation has the potential to reduce the severity of transmission issues. Data compression decreases the volume of transmitted data, aiding in reducing the usage of energy in sensor nodes and prolonging the lifespan of WSNs. However, traditional methods such as source and in-network coding tend to be computationally intensive and require prior global network information, rendering them impractical for WSNs. Many existing data-gathering schemes fail to utilize correlations in the sensed data, resulting in the delivery of a larger number of original packets. As illustrated in Figure 3A, nodes closer to the sink typically relay numerous packets from more distant nodes. For example, node #5 may transmit 11 packets, while each leaf node only passes 1. This imbalance in energy usage can quickly cause network breakdown.

Compressive sensing (CS) offers an innovative method for data aggregation in WSNs. The CS approach promises to cut network traffic without changing the data correlation structure. This approach sends compressed data over the network rather than the original data. The CS approach allows for the recovery of original sensor data readings at the sink, where recovering N sensor readings in the sink requires just M constant measurements ($M \ll N$). Due to the constant M, two conflicting effects may arise: data redundancy and data insufficiency. Both of these issues can diminish network transmission efficiency and compromise data recovery accuracy.

Therefore, compressive sensing (CS) has been developed into compressive data gathering (CDG), which offers an alternative technique for data collection in WSNs. It is regarded

as one of the most effective techniques for collecting sensed data on its way to the sink in WSNs. Additionally, it has low computational complexity and does not require prior or global information, making it ideal for resource-constrained WSNs. Plain-CDG [4], illustrated in Figure 3B, is the most ancient fundamental data-gathering approach that employs compressive sensing (CS) and network coding (NC) theory [5]. During each data-gathering epoch, each node sends a fixed number of packets, denoted as M. The Aggregation node is responsible for collecting and aggregating data from parent nodes before forwarding it towards the root node for further processing.

The authors of [6] proposed a technique based on Plain-CDG termed Hybrid-CDG, which is depicted in Figure 3C and aims to reduce the number of delivery packets. In the Hybrid-CDG scheme, nodes with degrees equal to or less than M employ Non-CDG to relay their measurements, while those with higher degrees generate M measurement packets with Plain-CDG. This method considerably minimizes redundancy; for example, leaf node #1 sends three packets in Plain-CDG but only one in Hybrid-CDG.

Figure 3D illustrates more efficient data collection systems that make use of random sparse measurements. Initially, M projection nodes are selected at random to collect M measurements; then, each projection node generates its sparse vector Φi and requests all nodes to submit their contributions. By acquiring all segments of a measurement ($y = \varphi_{i1}x_1 + \varphi_{i2}x_2 + \cdots + \varphi_{i11}x_{11}$), where x_j is the value obtained from node #j and φ_{ij} is its coefficient, the projection node delivers the final result to the sink node via the shortest routing channel. These sparse measurement-based CDG [7] schemes are more efficient than previous dense schemes because each measurement involves only a few nodes, allowing the rest to remain idle or in sleep mode to conserve energy. Therefore, sparse-CDG integrates sleep scheduling [8] to enhance energy conservation.

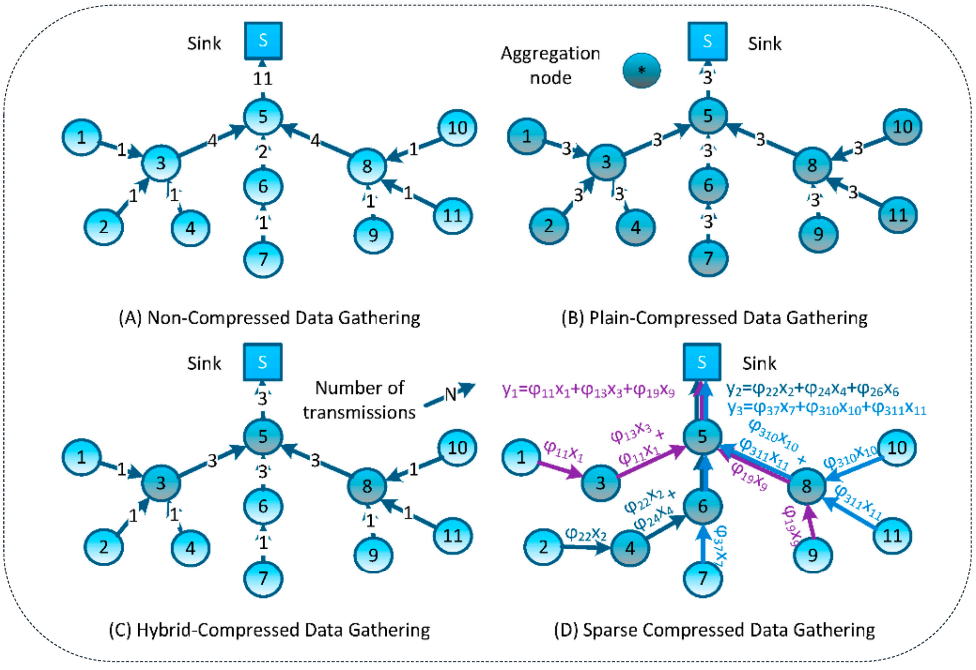

Figure 3. Non-CDG and typical CDG schemes [9]. Most of the existing wake-up/sleep scheduling studies for CDG are designed as centralized optimization problems, necessitating extra control for message exchanges and leading to uneven energy consumption between nodes and reduced WSN lifespans [10,11]. Additionally, most of the published CDG clustering techniques and routing methods are static and are unable to adapt to variations in the WSN's environment [12].

Therefore, a reinforcement learning-based dynamic clustering sleep scheduling algorithm (RLDCSSA-CDG) for compressive data gathering in WSNs is proposed to address

these issues. Reinforcement learning (RL) [13] allows the learning agent to adapt to environmental changes, making it a suitable intelligent algorithm for resource-constrained WSNs. The proposed RLDCSSA-CDG will be applied to the network through the 3F phases: foundation, formation, and forwarding phases.

Firstly, the network will be divided into clusters in a foundation phase. Each cluster is controlled by a unique cluster head (CH), which is promoted based on residual energy and distance to the sink node. Secondly, sensor nodes dynamically join clusters using the Upper Confidence Bound (UCB) [14] algorithm to enhance network performance and improve communication. Finally, after joining is formed, only a part of the nodes are selected to wake up just at the "right time" to sense data from the surrounding area by applying the sparse-CDG technique. Nodes that are not selected to participate in sparse-CDG can switch into sleep mode, which leads to a sleep scheduling algorithm.

The "right time" to wake up is at the nominal arrival time of the next periodic packet minus some safety margin to avoid packet loss because of waking up too late. On the other hand, the "right time" to go back to sleep mode (in case the expected packet does not arrive) is at the nominal arrival time of the next periodic packet plus some safety margin to avoid packet loss because of sleeping too early. The size of this safety margin directly determines the duty cycle (fraction of time in which nodes wake up) of the traffic periods. It has a direct influence on the forwarders' energy consumption as well as their packet losses. Therefore, this paper proposed a dynamic clustering technique based on an adaptive sleep scheduling algorithm to extend WSN lifetime and enhance energy efficiency, unlike most existing studies that use a static clustering based on the duty cycling technique, which cannot adapt to the variation in the WSN and incurs a tradeoff between packet delivery delay and energy saving. The main contributions of this work are summarized as follows:

- Dynamically clustering for WSN nodes is achieved by using the lightweight Upper Confidence Bound (UCB) algorithm, which improves adaptation in the WSN environment.
- Selecting the cluster head (CH) is carried out using a distributed method that considers the residual energy and distance among nodes rather than fixed CHs.
- Applying an adaptive wake-up/sleep scheduling algorithm by avoiding duty cycling is used to eliminate the tradeoff between energy savings and packet delivery delays.
- Sampling uniformity is included in the reward function of the RLDCSSA-CDG to evenly sample the sensed data for accurate reconstruction of the original data.

The rest of this paper is organized as follows: Section 2 offers a brief review of the background and related works on clustering techniques and sleep scheduling algorithms. Section 3 presents a detailed exploration of the proposed RLDCSSA-CDG algorithm in WSNs. Section 4 summarizes the complete proposed algorithm, Section 5 showcases the simulation results used to assess performance. Finally, Section 6 concludes the paper.

2. Related Work

A significant amount of literature focuses on improving the performance of CDG technology. In [15], a multi-objective evolutionary algorithm was applied to calculate the optimal number of CS measurements and determine the measurement matrix. In [16], a gray wolf optimization technique was used to develop an optimal sensing matrix that minimizes the mean square error of recovered data. Sleep scheduling and data compression can further decrease energy consumption in WSNs.

Sleep scheduling is commonly employed to tackle data redundancy resulting from densely deployed nodes. Its goal is to sustain WSN coverage and connectivity while minimizing the number of active nodes. This approach often complements other technologies to improve energy efficiency. For instance, ref. [17] presented a sleep scheduling strategy for opportunistic routing in one-dimensional queue networks, where sleep intervals for

nodes were determined based on the flow rate and remaining energy. Similarly, ref. [18] focused on reducing energy consumption from idle listening during sleep scheduling by organizing nodes into pairs, allowing them to alternate between active and sleep modes based on the traffic rate and their remaining energy, but total data transmission and data recovery accuracy are not considered.

Lastly, ref. [19] utilized deep reinforcement learning, adaptive learning automata, and particle swarm optimization to select sleep and active nodes. An RL-based sleep scheduling algorithm for CDG (RLSSA-CDG) was introduced in [20] to achieve a balanced energy expenditure among nodes and extend the lifespan of WSNs. These studies confirmed that various sleep scheduling methods effectively promote energy efficiency in WSNs. However, they overlooked the challenges posed by high-dimensional data transmission. Combining sleep scheduling with data compression can further reduce energy consumption in WSNs. Numerous studies have aimed to improve CDG techniques, particularly through different clustering technologies. Clustering is beneficial for traffic load balancing and enhancing network energy efficiency, especially in large-scale WSNs.

Thein et al. [21] modified the probability for selecting cluster heads (CHs) based on each node's residual energy, enhancing network lifetime by 40–50% while considering fixed CH values. In the Hybrid Clustering-based Data Collection Scheme (HCDCS) in [22], sensor nodes are grouped based on data correlation, which reduced the number of necessary CS measurements, but latency is not considered. In [23], a seed estimation algorithm created an adaptive dynamic random seed to generate an optimal random measurement matrix with minimal reconstruction error. References [24,25] designed clustering methods that improved the CH selection criteria by considering the distance to the sink, residual energy, intra-to-inter distance ratios, compression ratios, and node density. References [26,27] addressed load balancing in clustering-based CDG by grouping nodes according to distribution density or equal energy consumption.

In [28], a reinforcement learning-based dynamic clustering algorithm (RLDCA) was introduced to reduce data transmissions and energy consumption in WSNs. The Minimum Distance-based Clustering (MDC) method [29] was developed without CS and RL considerations that affect the whole network lifetime. Most of the existing protocols highlighted inefficiencies in clusters that are too small and proposed a technique for balancing cluster sizes to optimize CDG utilization. In general, challenges in a clustering CDG involve deciding the number of clusters, selecting cluster heads (CHs), forming clusters, and planning data transmission methods. The main goals focus on reducing data transmission, improving energy efficiency, balancing traffic load, and ultimately extending the network's lifespan. Various studies in the literature may address one or more of these issues. Additionally, many of these clustering methods are static and based on minimum distance, making them less adaptable to fluctuations in sensing data or environmental conditions. Table 1 summarizes the literature survey of related protocols for performance comparison.

To summarize, the proposed RLDCSSA-CDG is a dynamic clustering method integrated into sleep scheduling based on the RL algorithm that offers improved adaptability, which is more suitable for WSN time-varying. It is designed to perceive the residual energy of nodes, achieving a balanced energy expenditure and prolonging the overall network lifetime. In other words, energy is unnecessarily wasted if no packets are exchanged. The goal of sleep/wake-up scheduling is to minimize this waste by optimizing the awake time of sensors. The proposed algorithm looks at how to adjust the sleep and awake time ratio in each cycle based on an adaptive wake-up/sleep scheduling.

Table 1. Summarization of the related protocols.

Protocol	Latency	Data Transmission	Data Recovery	Energy Consumption	Network Lifetime
RLDCSSA-CDG (proposed protocol)	Medium	Low	High	Low	High
RLSSA-CDG [20] (Wang, X., H. Chen, and S. Li, 2023)	Medium	Medium	Medium	Low	High
RLDCA [28] (Wang, X., H. Chen, and J.M. Barcelo-Ordinas, 2022)	Low	Medium	Medium	Low	Medium
HCDCS [22] (Lin, C., et al.,2020)	Low	Medium	Medium	High	Medium
MDC [29] (Bai, T., et al., 2019)	Low	High	Medium	High	Low
Sparse-CDG [30] (H.F. Zheng, F. Yang, X.H. Tian, X.Y. Gan, X.B. Wang, 2015)	Low	High	Low	High	Low

3. Proposed RLDCSSA for Compressive Data Gathering

Extending the lifetime of wireless sensor networks (WSNs) is a crucial challenge due to limited energy resources. This paper introduces a reinforcement learning-based dynamic clustering of sleep scheduling algorithm (RLDCSSA-CDG) for compressive data gathering in WSNs, aimed at prolonging network lifetime. The RLDCSSA-CDG operates in rounds, with each round divided into three main phases: foundation, formation, and forwarding phases as shown in Figure 4. Firstly, the WSN architecture is designed in the foundation phase and the cluster heads (CHs) nodes will be determined. Secondly, the formation phase performs dynamic clustering based on the Upper Confidence Bound (UCB) algorithm. Finally, an adaptive sleep scheduling algorithm will be applied to the sensor nodes to improve network lifespan. In the next subsections, each phase will be discussed in more detail.

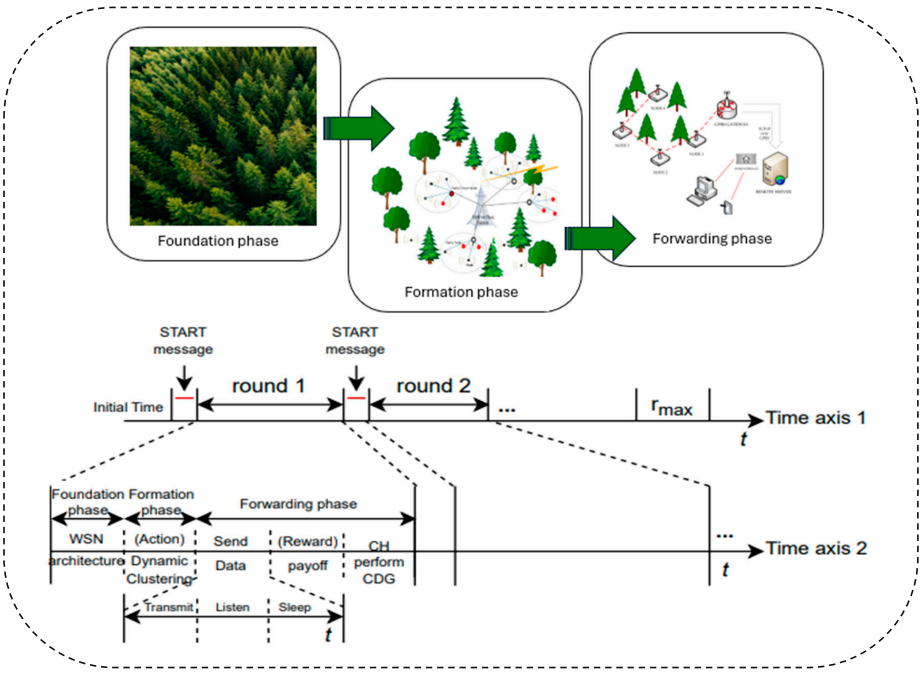

Figure 4. The proposed RLDCSSA-CDG phases.

3.1. Foundation Phase

Consider N static sensor nodes, represented by the set N = {s_1, s_2, s_3,· · · , s_N}, deployed randomly and uniformly in a two-dimensional area Z. These battery-powered nodes are non-rechargeable and continuously monitor physical phenomena such as temperature, humidity, and pressure. All nodes remain static and periodically collect ambient data. Among various methods to reduce energy consumption in WSNs, clustering is a well-recognized technique. This technique allows for efficient energy use and helps to prevent collisions. Sensors are organized into clusters, each with one cluster head (CH) responsible for data aggregation.

The proposed algorithm operates in rounds, with the CH changing each round based on an election probability, ensuring that all nodes in the cluster have an equal chance of being selected as the CH. The clustering design incorporates factors such as residual energy and distance to the sink in the criteria for selecting the cluster head (CH). The equiprobable CH election process can result in the selection of a CH with low residual energy, which may deplete faster than one with higher energy levels.

To tackle this issue, the residual energy of each node is incorporated into the CH election probability equation, ensuring that nodes with greater energy have a higher chance of being selected as the CH. In the first round, clusters and CHs are established using the standard LEACH algorithm [31]. After data transfer, each node consumes varying amounts of energy, making the network heterogeneous. Consequently, in subsequent rounds, the CH is elected using a modified threshold equation based on residual energy, as shown in Equation (1):

$$\mathrm{T(n)} = \begin{cases} \frac{P}{1-P\left(r \bmod \frac{1}{P}\right)} \times \frac{E_{residual}}{E_{initial}} K_{opt} & for\ all\ n\ \acute{\mathrm{U}}\ G \\ 0 & ,otherwise \end{cases} \quad (1)$$

where P denotes the desired proportion of cluster heads (CHs), r represents the number of rounds, and G includes the nodes not selected as CHs in the last $1/p$ rounds. $E_{residual}$ is the remaining energy level of the node, and $E_{initial}$ is the initial energy level. The distance to the sink is a key factor in CH selection, as it influences the data transmission distance; shorter distances are preferred for CHs. This parameter is integrated into the threshold formula as outlined. The optimal number of clusters, k_{opt}, can be expressed as in Equation (2):

$$K_{opt} = \sqrt{n/2\pi}\sqrt{\frac{E_{fs}}{E_{amp}d^4(2m-1)E_0 - mE_{DA}}}M \quad (2)$$

where M represents the network diameter, and d denotes the distance between the sending and receiving nodes. E_0 is the initial energy provided to each node, E_{fs} is the energy dissipation in the free space model, E_{amp} is the energy dissipation in the power amplifier, and E_{DA} is the energy used in data aggregation. Each sensor node generates a random number γ_i uniformly distributed in the range [0, 1], as shown in Figure 5. If γ_i for the i-th sensor node (SN) is less than threshold T(n) and no other CHs are within its communication range, the SN elects itself as the CH for the current round.

Conversely, if other CHs are available, the one with the highest energy reserve is chosen as the final CH, while the others remain regular SNs. Each CH selected based on the T(n) threshold broadcasts a control message that includes its remaining energy. Upon receiving this message, a CH can declare itself the final CH if it has more energy than the others within its communication range. The collection of CHs is referred to as the action set, denoted as $A = \{ch_1, ch_2, \ldots, ch_p\}$.

However, re-clustering consumes considerable energy in constrained sensor networks. Therefore, SNs need to implement an efficient algorithm for CH selection based on energy levels to improve network stability and lifespan. To minimize the need for re-clustering, we propose an energy-threshold-based algorithm that delays CH selection, allowing the same topology to be maintained for several rounds without the overhead of control exchanges. CH selection is triggered only when a CH's energy falls below a defined threshold, which is set to ensure proper energy balancing among all SNs. This threshold, determined through simulations, is set at 90% of the CH's initial energy at the start of a round [32]. Thus, each CH monitors its energy level, and if it drops below 90%, it initiates the new CH election process.

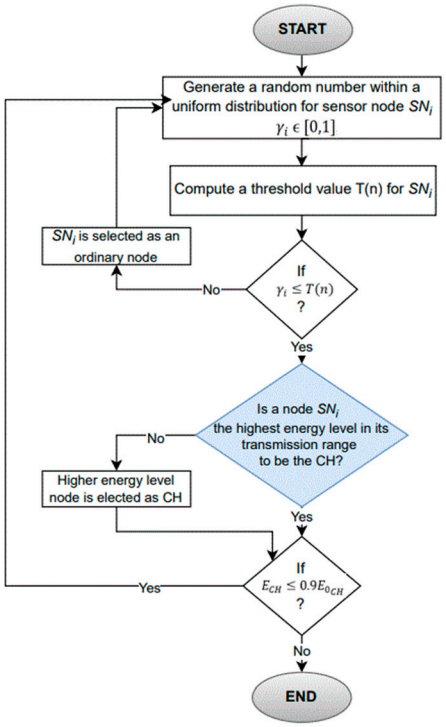

Figure 5. Flowchart of the foundation phase.

3.2. Formation Phase

Sensing data in the network varies over time, leading to developing a dynamic clustering scheme incorporating reinforcement learning (RL) to improve network adaptability. In this dynamic clustering framework, nodes reselect their clusters at the start of each round. Each node functions as an independent RL agent, employing the RL algorithm under uniform scheduling. During each round, node SN_i decides to join the j-th cluster and transmits its sensing data to the cluster head ch_j. In return, *SNi* receives a reward for its action, which is stored in its memory and accumulated as a learning experience. This constitutes an online learning scheme that does not require pre-existing training data.

Therefore, the RLDCSSA-CDG algorithm was proposed to create a dynamic clustering algorithm for compressive data gathering (CDG) by using RL principles. The algorithm utilizes the features of CDG to balance traffic load and extend the lifespan of the WSN. It begins by using CDG to minimize data transmissions and then incorporates the RL algorithm to improve the overall performance of CDG. According to compressive sensing (CS) theory, the sensing data from cluster members (CMs) in the i-th cluster, denoted as $xi = [x_{1i}, x_{2i}, x_{3i} \cdots, x_{L_i i}]^T$, is sparse or compressible with a sparsity level of K_i. The data xi,

which have a length of L_i, can be compressed to y_i using M_i measurements, as expressed in Equation (3):

$$y_i = \varnothing_i x_i = \varnothing_i \Psi_i s_i = A_i s_i \quad (3)$$

where Φ_i is a Gauss random matrix generated by the CH, and Ψ_i is a $L_i \times L_i$ discrete cosine transformation basis that converts a compressible signal into a sparse signal in the transformation domain. The matrix $A_i = \Phi_i \Psi_i$ is known as the sensing matrix or measurement matrix in CS. For A_i to satisfy the Restricted Isometry Property (RIP), it must meet the conditions outlined in Equation (4):

$$M_i \geq C \,.\, K_i \,.\, \log \frac{L_i}{K_i} \quad (4)$$

where C is a constant and K_i denotes the sparsity of x_i. If these conditions are satisfied, x_i can be recovered from M_i measurements in yi with a high probability close to 1. Typically, when Φ_i is a random matrix, $M_i = 3K_i \sim 4K_i$ is sufficient for accurate recovery. The dynamic number of compressive sensing (CS) measurements M is calculated based on the intra-cluster data sparsity during each round of data gathering. This method ensures both efficient data transmission and accurate data recovery. The proposed scheme takes advantage of the data correlation among nodes; if the correlation among the original data is strong, fewer CS measurements are required for accurate recovery.

In the RLDCSSA-CDG, each node selects a cluster with strong data correlation to join. The level of data correlation can be quantified using the Pearson correlation coefficient, as expressed in Equation (5):

$$\rho_{xy} = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \overline{x})^2 . \sum_{i=1}^{n} (y_i - \overline{y})^2}} \quad (5)$$

where ρ_{xy} represents the correlation coefficient between two data sets $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, while $\overline{x}$ and $\overline{y}$ denote the mean values of x and y, respectively. Data sparsity is closely related to the correlation within the data set. Thus, sensor nodes exhibiting strong correlations in their sensing data are grouped into a cluster. This allows for more compact compression of intra-cluster data, significantly reducing data transmissions across the entire network. The objective function of the proposed RLDCSSA-CDG is to minimize data transmissions in the WSN and prolong the network's lifetime. This can be formulated as follows in Equation (6):

$$\min \sum_{r=1}^{r_{max}} \sum_{i=1}^{P} (L_i + M_i) \quad (6)$$

where L_i represents the intra-cluster data transmissions in the i-th cluster, M_i denotes the data transmissions of the CH chi in the i-th cluster, p is the number of clusters in the network, and r_{max} indicates the maximum number of running rounds during the WSN's lifetime. We utilize the classical energy consumption model from [33]. When sending a data packet of l bit, the energy consumption E_{Tx} for a node encompasses the energy used by processing circuits and transmitting amplifiers. This can be expressed as in Equation (7):

$$\begin{gathered} E_{Tx}(l,d) = E_{Tx-elec}(l) + E_{Tx-amp}(l,d) \\ = \begin{cases} l\left(E_{elec} + \varepsilon_{fs} d^2\right) & d < d_0 \\ l\left(E_{elec} + \varepsilon_{mp} d^4\right) & d \geq d_0 \end{cases} \\ E_{Rx}(l) = E_{Rx-elec}(l) = l E_{elec} \\ d_0 = \sqrt{\frac{\varepsilon_{fs}}{\varepsilon_{mp}}} \end{gathered} \quad (7)$$

where E_{elec} = 50 *nJ/bit* is the energy consumed by the processing circuits to transmit or receive a bit of data, d is the distance between the sender and receiver, and d_0 is the distance threshold for free space. When $d < d_0$, the free space channel model is used, and the energy consumed by the amplifier to send 1 bit of data is expressed as follows: $E_{amp} = \varepsilon_{fs} d^2$, where ε_{fs} = 10 $pJ/bit/m^2$. When $d \geq d_0$, the multipath fading channel model will be applied and the energy consumed by the amplifier to send a bit of data is given by $E_{amp} = \varepsilon_{mp} d^4$, where ε_{mp} = 0.0013 $pJ/bit/m^4$.

Reinforcement learning (RL) mimics human behavior learned through interactions with the environment. This allows sensors and sink nodes to observe and execute optimal actions for improved network and application performance. The agent RL is not explicitly instructed on which action to take; rather, it autonomously selects actions and receives rewards based on its choices.

The agent's goal is to maximize the total reward over time. A well-designed reward scheme is crucial for guiding the agent's behavior. The RL process involves the agent observing the environment to obtain a state vector S, selecting an action from the action set A, and then receiving a reward r based on a reward function R. To maximize the total reward, a value function is used to estimate the average reward of each action in A.

However, this value function is often unknown and must be evaluated through continuous trials. An agent can use the evaluated value function to choose the action with the highest estimated value, but this estimate may not accurately reflect the true value. Therefore, the agent must also explore actions that may not seem to have the highest value. This balance between exploiting known actions and exploring new ones is known as the exploration dilemma.

In this paper, the task assigned to the agents is to select a cluster to join, which is a multi-choice problem effectively addressed by the Upper Confidence Bound (UCB) algorithm [14]. The UCB algorithm provides a suitable balance for the exploitation dilemma in RL. The lightweight UCB algorithm serves as a guideline for the agents in making decisions regarding their actions. The strategy for action selection is based on Equations (8) and (9):

$$A_t = \binom{\text{argmax}}{a} \left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right] \tag{8}$$

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} r_i \cdot I}{\sum_{i=1}^{t-1} I},\ I = \begin{cases} 1 & \text{when the selected action is "}a\text{"} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

In Equation (8), A_t represents the action selected at time step t, and $Q_t(a)$ is the evaluated value function for action a, as defined in Equation (9). Here, $c > 0$ is a constant that controls the level of exploration, ln t denotes the natural logarithm of t, and $N_t(a)$ indicates how many times action a has been selected prior to time t. In Equation (9), r_i represents the corresponding reward at time i. In this context, the design of the corresponding components state, state vector, action set, and reward function focuses on intra-cluster data correlation and the distances between nodes and cluster heads (CHs).

The RLDCSSA-CDG is implemented in a distributed manner, with each sensor node in the network acting as an independent RL agent. This distributed algorithm reduces the complexity of the state and action vectors and minimizes communication overhead between the sink and the sensor nodes. The agents interact with the same environment and adhere to the same learning process, meaning they share the same state vector S, action set A, and reward function R. These components are defined as follows:

- **The state vector S** represents an agent's observations of the environment and is essential for calculating rewards. In RLDCSSA-CDG, the state vector is defined as $S = [\rho_i, d_i]$, where ρi indicates the data correlation between the agent and the i-th

cluster (calculated using Equation (5)), and di represents the distance from the agent to the cluster head (CH) of the i-th cluster.

- **The action set A** consists of a total of p clusters based on the foundational state. At the start of each round, each node selects a cluster to join. Thus, the action set A is formulated as $A = \{ch_1, ch_2, \cdots, ch_p\}$, where ch_i is the cluster head of the i-th cluster. When an agent takes action ch_i from A, it signifies that the node joins the i-th cluster and transmits its sensing data to the cluster head ch_i.
- **The reward function R** incorporates two elements from the state vector S and is defined as follows, where $0 < \alpha < 1$ is a constant that adjusts the emphasis on the two elements in S. A higher value of α indicates that nodes will prioritize data correlation more when selecting a cluster. Here, r_c and r_d are the rewards based on ρ_i and d_i in S, respectively. To simplify the computational complexity for nodes, a linear and relative reward function model is used. First, the p actions in action set A are sorted in descending order of ρ_i and ascending order of d_i. Then, the normalized reward values r_c and r_d are assigned to the p actions in A based on this order. The reward values for r_c range from 1 to -1, with the interval between two adjacent increments defined as $\Delta = 2/(p-1)$.

3.3. Forwarding Phase

Achieving a long node lifetime in wireless sensor networks (WSNs) is a significant research concern due to the limited energy available to each node. One effective strategy for prolonging node lifetime is the sleep scheduling algorithm, which enables nodes to switch into a sleep state whenever possible. In the sleep state, certain hardware components (i.e., transceiver) are powered down to reduce energy consumption. Most existing algorithms do not account for fluctuations in node residual energy, causing uneven energy expenditure across nodes and ultimately shortening the network's lifespan. The proposed algorithm aims to balance energy expenditure among nodes, thereby extending the overall lifespan of the network. The transitions between the sleep and active states of nodes can be modeled as a finite Markov decision process (MDP) [34], framing this problem as a reinforcement learning (RL) task. The primary objective of the RL algorithm is to find an optimal policy that maximizes cumulative rewards over time. In this paper, we define the mean reward of an action as its value, and the state-action value function Q(s, a) is used to evaluate actions. An optimal policy π can be defined as the policy that yields the maximum value, as expressed in Equation (10):

$$Q*(s,a) = max_{\pi} Q_{\pi}(s,a) \tag{10}$$

where $Q(s, a)$ represents the value of taking action a in state s. The best policy for environments with a known model can be determined by solving the Bellman optimality equation [35]. The WSNs' environments are often unknown or dynamic. Hence, we utilize the model-free Q-learning algorithm, which learns directly from raw experience without requiring a model of the environment's dynamics. The Q-value function will be updated based on previous estimates without needing to wait for the final outcome. This algorithm is especially useful in environments where the dynamics are not fully understood. The iteration equation for updating the Q-value in Q-learning is expressed as follows in Equation (11).

$$Q(s_k, a_k) = (1-\alpha)Q(s_k, a_k) + \alpha[r_{k+1} + \gamma max_A Q(s_{k+1}, a_{k+1})] \tag{11}$$

where s_k and s_{k+1} represent the states at time k and $k+1$, respectively. The set of all possible actions is denoted by A. The actions taken at time k and $k+1$ are represented by a_k and a_{k+1}.

The immediate reward received after taking action a_k in state s_k is denoted by r_{k+1}. The learning parameter, α, controls the speed of convergence; a higher α leads to faster learning but can also cause instability. The discount factor, $\gamma \in [0, 1]$, balances immediate and future rewards. A value closer to 1 emphasizes future rewards, while a value closer to 0 focuses on immediate rewards. The Q-learning model uses discrete, finite states to allow clear action decisions based on a shared Q-table across the network. This Q-table serves as the "brain" of the agent, storing estimated values for each action in every state. Each node maintains one row of this Q-table. For illustration, the adaptive wake-up/sleep scheduling algorithm operates within a single time slot through three integrated algorithms: Algorithms 1–3 as shown in Figure 6.

Figure 6. The adaptive wake-up/sleep scheduling algorithm.

Algorithm 1 is initiated as illustrated in Figure 7. It begins at line 1 for the first timeslot. Otherwise, it resumes from line 3. The learning rate controls how much new information influences existing knowledge, with values from 0 to 1. A value of 0 means no learning; a value of 1 means only the most recent information is considered. The discount factor, also ranging from 0 to 1, determines the focus on future rewards; 0 focuses solely on immediate rewards, while values close to 1 emphasize long-term rewards [14]. At the start of each time slot, nodes (line 4) decide their operational mode by selecting an action based on a probability distribution of available actions (e.g., transmit, listen, sleep). Initially, the probability for each action is set equally (e.g., 1/3 for three actions). This distribution updates based on previous outcomes. If the selected mode is "transmit," nodes determine when to send packets during the time slot, as described in Algorithm 2.

After transmission, nodes receive a payoff and transition to a new state. The Q-value for the chosen action is updated based on the received payoff and the maximum Q-value in the new state (line 7), combining old estimates with new rewards. In lines 8–10, if the selected mode is not "sleep," nodes estimate the probability distribution of their neighbors' actions based on interactions in the current slot. This updates the policy π(s, a) for each action [36]. In lines 11–14, if the node chooses "sleep," the policy is updated based on the average payoff, calculated from the Q-value and the probabilities of selecting each action. The average payoff is computed by multiplying the Q-value of an action by its probability. The probability of selecting an action is adjusted based on the difference between the Q-value and the average payoff: if $Q(s, a)$ > average payoff, the probability of action a increases; otherwise, it decreases (line 13). In line 15, the probability distribution π(s) is normalized so that $\sum_{(a \in A)} \pi(s, a) = 1$, ensuring each π(s, a) remains within [0, 1], as described in Algorithm 3. Finally, in line 16, the learning rate ξ is decayed over time to ensure convergence. At the start of the next time slot, the algorithm loops back to line 3 of Algorithm 1, continuing the learning and decision-making process [37].

$\xi, \delta, \zeta, \epsilon$	Learning rates
γ	Discount factor
s	State
A	An action
N	No. of available actions
Q	Reinforcement Value for a node
$Q(s, a)$	The Reinforcement Value for a node to take action a in state s
π	Policy for a node
$\pi(s, a)$	Probability for a node to select action a in state s
P	Payoff
$\bar{p}$	Average payoff
k	Current time slot
X	Probability distribution
m	No. of sub-slots
C_0	mapping center
Δ	mapping lower bound

Figure 7. Parameters of an adaptive wake-up/sleep scheduling algorithm.

Unfortunately, When the duration of a time slot exceeds the time required to transmit a packet, nodes need to strategize when to transmit within that slot. There are two main options: First, if a time slot is long enough to accommodate m packets, it can be divided into m sub-slots, with each sub-slot lasting as long as it takes to transmit one packet. Second, nodes must select one of the available sub-slots for their transmissions. Two intuitive strategies for choosing a sub-slot are Random Selection (RS) and Continuous Attempts (CA). In Random Selection (RS), the node randomly chooses a sub-slot to transmit the packet. However, this can lead to the inefficient use of time and energy, as the chosen sub-slot may not be optimal for transmission. In Continuous Attempts (CA), the node starts transmitting at the beginning of the time slot and continues until the packet is successfully sent or the time slot ends. This method can waste considerable energy, especially if the node keeps trying to transmit unsuccessfully in rapid succession. To mitigate these inefficiencies, Algorithm 2 is designed to determine the optimal time for packet transmission within the time slot.

In Algorithm 2, the node selects a sub-slot based on a probability distribution, x, over the available sub-slots (line 3). After choosing a sub-slot, the node observes the resulting payoff, p, for that choice (line 4), which reflects whether the transmission was successful or not. The Q-value for each sub-slot is then updated using the observed payoff and the current probability distribution x. This update helps the node learn by adjusting the value of each sub-slot based on its performance. Next, the probability distribution x is modified according to the updated Q-values of the sub-slots (line 5). This modification uses the ϵ-greedy exploration strategy to balance exploration and exploitation. The best sub-slot is chosen with a probability of $1 - \epsilon + (\epsilon/m)$, while each of the remaining $m-1$ sub-slots is chosen with a probability of ϵ/m. The updated probability distribution is normalized to ensure that the sum of the probabilities equals 1 (line 6). Combining ϵ-greedy [38] exploration with Q-learning makes the decision-making process more efficient and adaptive over time.

Algorithm 1: Sleep/Wake-up Scheduling of a Node

Let ξ and δ be the learning rates and γ be the discount factor;

For each action, initialize value function Q to 0 and policy π to $\frac{1}{n}$, where n is the number of available action;

repeat

Select an action a in current state s based on policy $\pi(s,a)$;

if the selected action a is transmit **then**

the node determines when to transmit the packet in the time slot; /* **Algorithm 2***/

Observe payoff *P* and next state s', update Q-value

$Q(s,a) \leftarrow (1-\xi)Q(s,a) + \xi(P + \gamma max_{a'}Q(s',a'))$;

if the selected action a is not sleep **then**

based on the updated Q-value, approximate the policy of the neighbor that interacted with the node in the current time slot;

based on the approximation, for each action $a \in A$, update the node policy $\pi(s,a)$;

else

calculate the average payoff

$\bar{P}(s) \leftarrow \sum_{a \in A} \pi(s,a)\, Q(s,a)$;

For each action $a \in A$ **do**

$\pi(s,a) \leftarrow \pi(s,a) + \delta(Q(s,a) - \bar{P}(s))$;

$\pi(s) \leftarrow$ Normalize($\pi(s)$); /***Algorithm 3***/

$\xi \leftarrow \frac{k}{k+1}.\xi$;

$s \leftarrow s'$

Until *the process is terminated;*

To ensure that a probability distribution is valid, we use proportion-based mapping. This method is crucial for preserving the integrity of learned knowledge while normalizing probabilities. Proportion-based mapping adjusts any invalid probability so that it falls within the range [0, 1]. The goal is to keep the normalized probabilities as close as possible to the original unnormalized ones. The normalization process is handled by a function called Normalize(), which is outlined in pseudocode in Algorithm 3. By using proportion-based mapping for normalization, this algorithm ensures valid probability distributions and enhances the nodes' ability to adapt. This leads to better decision-making for sleep/wake cycles, making these nodes smarter and more responsive than traditional methods.

Algorithm 2: Node Determines When to Transmit a Packet in a Time Slot

Let ξ and $\in$ be the learning rates;
For each sub-slot in the current time slot, initialize Q-value to 0 and the probability for selecting each sub-slot is initialized to $x_i = \frac{1}{m}$, where $1 \leq i \leq m$ and m is the number of sub-slots;
Select a sub-slot in current time slot based on the probability distribution over sub-slots $x = \langle x_1, \ldots, x_m \rangle$;
Observe payoff P and update Q-value for each sub-slot,

$$Q_i \leftarrow Q_i + x_i.\xi \cdot \left(P - \sum_{1 \leq i \leq m} x_i Q_i \right);$$

Update x_i for each sub-slot,

$$x_i = \begin{cases} (1 - \in) + (\in / m), & if\ Q_i\ is\ the\ highest \\ \in / m, & otherwise \end{cases}$$

$x \leftarrow$ Normalize(x);

Algorithm 3: Normalize()

Suppose that in state s, there are m available actions, i.e., $a_1, a_2, \ldots a_m$;
Let d = $min_{1 \leq k \leq m}\ \pi(s, a_k)$, mapping center $c_0 = 0.5$ and mapping lower bound $\Delta = 0.0001$;
if d $< \Delta$ **then**
$\rho = \frac{c_0 - \Delta}{c_0 - d}$;
for k = 1 to m **do**
$\pi(s, a_k) \leftarrow c_0 - \rho \cdot (c_0 - \pi(s, a_k))$;
for k = 1 to m **do**
$r \leftarrow \sum_{1 \leq K \leq m} \pi(s, a_k)$;
$\pi(s, a_k) \leftarrow \frac{\pi(s, a_k)}{r}$;
return $\pi(s)$;

Finally, the CHs perform CDG and transmit the sensed data to the sink node in the forwarding phase as shown in Figure 8. After all measurements are processed, the sink verifies the integrity of the received data and broadcasts the START message for the next round. This data reconstruction can be formulated as a convex optimization problem, represented in Equation (12):

$$\min_{\theta \epsilon R^N} \| \theta \|\ l_i \text{ subject to } y = \Phi X = \Phi \Psi \theta \tag{12}$$

where θ is a sparse vector representing the transformation of the compressive data set X under an orthogonal sparse basis ψ. If X is a sparse vector, then ψ is the identity matrix. Therefore, the proposed RLDCSSA-CDG is an efficient algorithm designed for efficiency in a resource-limited environment, balancing performance with computational feasibility.

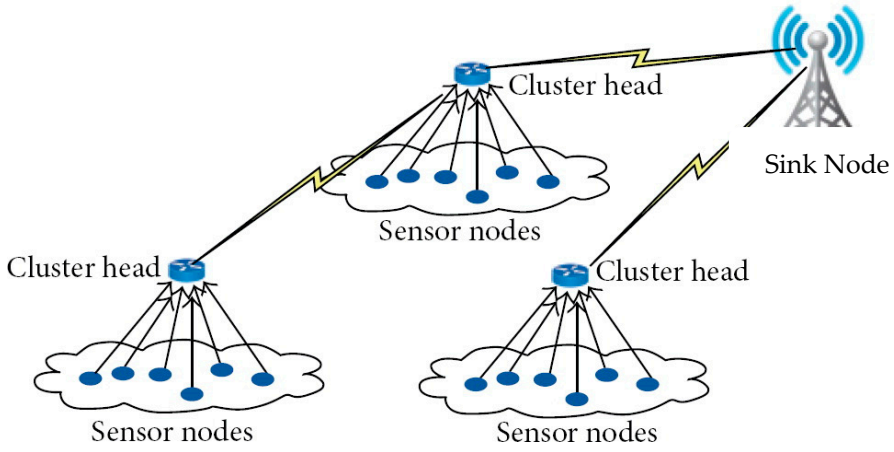

Figure 8. Forwarding phase [39].

4. The RLDCSSA-CDG Algorithm Summarization

The overall RLDCSSA-CDG algorithm is summarized in this section. The proposed algorithm is processed through 3F phases: foundation, formation, and forwarding phases. The proposed algorithm is illustrated as described in Figure 9. Firstly, the network architecture will be designed, and the cluster heads (CHs) will be selected in the foundation phase as depicted in Figure 5. Then, sensor nodes join the clusters dynamically in the formation phase based on the UBC algorithm. The dynamic clustering technique operates in rounds, with the maximum iteration number denoted as r_{max}. Key components include the initial action set A, the reward table R, and the Action Selection Value (ASV) table At. Each round begins with a START message from the sink. Since agents have no prior experience in the initial round, the reward and ASV tables for all actions are initialized to zero. Each agent randomly selects an action from the action set A and implements the RL strategy. In subsequent rounds, the agent identifies the optimal action based on the experiences recorded in the reward and ASV table. Then, the rewards are calculated, and the ASV table is updated.

In other words, computation resources in the proposed RLDCSSA-CDG are primarily consumed by the following operations: action selection involves p comparison operations based on Equation (8), where agents evaluate the potential actions to determine the best choice; the reward computation for r_c requires sorting p values, leading to a computational overhead of O(p log p); and the complexity for computing $Q_t(a)$ in Equation (9) is O(p). After receiving all measurements, the sink generates the measurement matrix based on the vector path to recover the original data. Subsequently, it broadcasts the START message for the next round. Generally, this online learning algorithm is designed for efficiency in a resource-limited environment, balancing performance with computational feasibility.

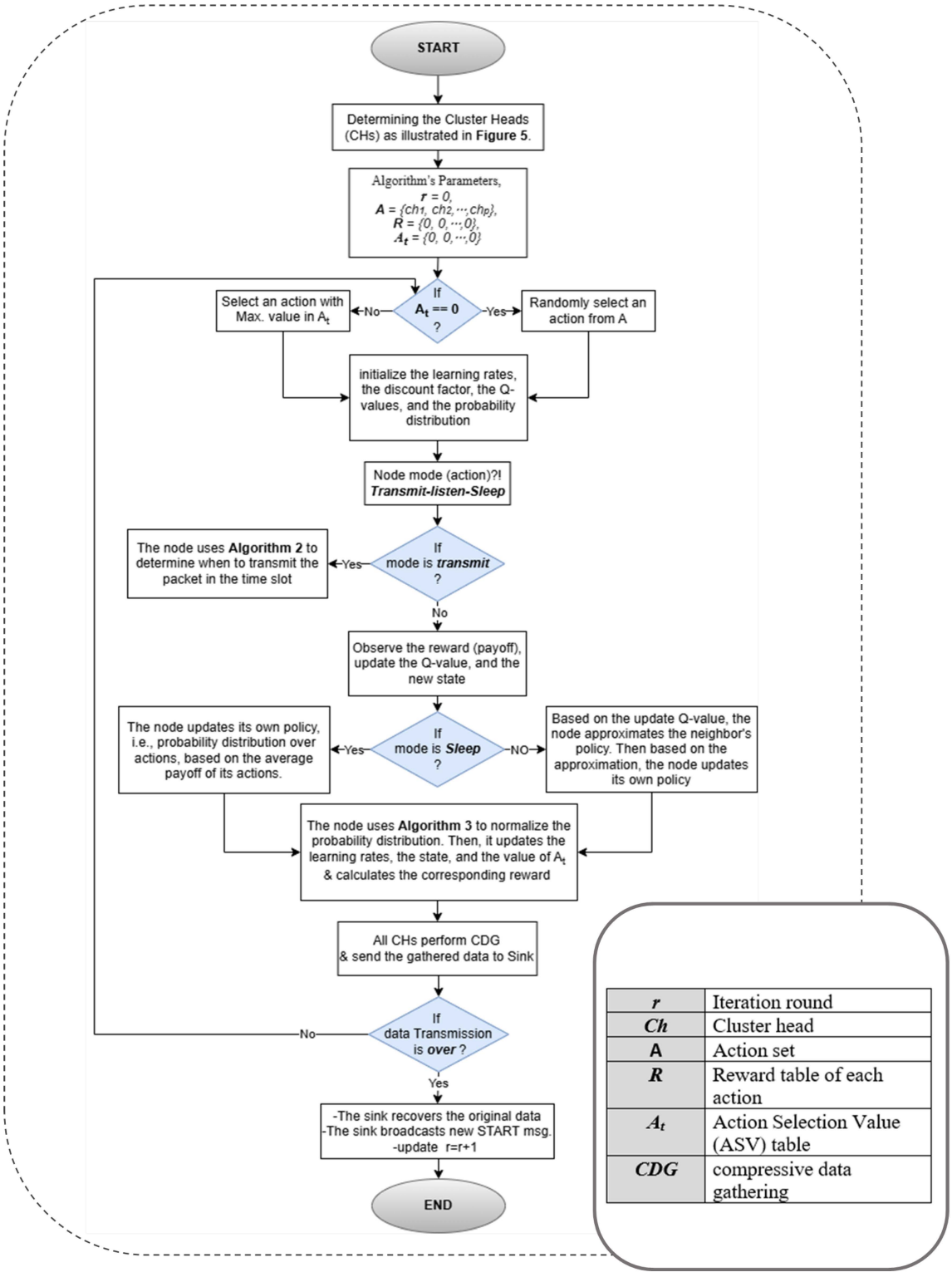

Figure 9. Flowchart of the proposed RLDCSSA-CDG.

5. Experimental Results and Analysis

In this section, we describe the simulation experiments designed to assess the performance of the proposed RLDCSSA-CDG algorithm. The simulations are performed using MATLAB and use the following parameters: A total of 512 sensor nodes are evenly and randomly distributed within a monitoring area of 2000 m × 1000 m, referred to as area Z. The sink is located at the center of this area, divided into clusters, supervised by a cluster

head (CH). The communication radius for CHs is set at 280 m, while sensor nodes have a communication radius of 140 m. Each sensing data packet has a length of 500 bits.

Detailed parameters related to the energy consumption model and the Q-learning algorithm are provided in Table 2. We will compare the performance of the RLDCSSA-CDG algorithm against two other algorithms: reinforcement learning-based dynamic clustering algorithm (RLDCA) for CDG [28] and reinforcement learning-based sleep scheduling algorithm (RLSSA-CDG) for CDG [20].

Table 2. Simulation environment values.

Parameter	Value
Monitoring area Z	2000 m × 1000 m
The number of nodes	N = 512
Node's initial energy	E0 = 0.5 J
Nodes' communication radius	RN = 140 m
CHs' communication radius	RCH = 280 m
Length of a data packet	500 bits
Threshold distance in free space	d0 = 88 m
Energy used for (transmit)	81 mw
Energy used for (listen)	30 mw
Energy used for (sleep)	0.003 mw
Time-to-live (TTL)	15
Discount rate	$\gamma = 0.65$

The proposed algorithm does not use a duty cycle, so we set the time slot to 8 ms and the full sleep/wake-up interval to 1 s. We employ learning rates to update our learned knowledge. In contrast, algorithms that do not use reinforcement learning do not require updates and thus do not need learning rates. Each simulation case is run 200 times, with each run lasting 5000 s. At the start of each time slot, each node generates a packet based on a predefined probability called the packet generation probability. This influences the state of each node, which is determined by the number of packets in its buffer. Packet expiry time follows an exponential distribution. The average packet size is 100 bytes, with the actual size following a normal distribution with a variance of 10.

This setup allows for the thorough testing of the RLDCSSA-CDG algorithm's performance against established benchmarks, ensuring a comprehensive evaluation of its effectiveness in a dynamic wireless sensor network environment. The simulation experiments are designed to evaluate the performance of the proposed RLDCSSA-CDG algorithm, focusing on various packet generation probabilities and key performance metrics, which are ξ, δ, ζ, ϵ set as 0.2, 0.4, 0.6, and 0.8, respectively. These probabilities help to assess the performance of the algorithms under different conditions regarding the number of transmitted packets. The performance of the proposed algorithm is evaluated using six quantitative metrics: average energy consumption, total data transmission, data recovery accuracy, average delivery latency, and network lifetime longevity.

5.1. Average Energy Consumption

Energy efficiency is a primary objective in various WSN technologies. Implementing sleep scheduling allows nodes to enter energy-saving sleep modes, thereby reducing overall energy expenditure. The average energy consumption is determined by dividing the total energy used by the number of nodes in the network during a simulation run. Figure 10 illustrates that the simulation results confirm the effectiveness of the proposed algorithm. Both RLDCSSA-CDG and RLSSA, which incorporate reinforcement learning (RL) with a sleep scheduling algorithm, demonstrate significantly lower energy expenditures compared to the RLDCA for CDG, which does not utilize a sleep scheduling algorithm. The advantage

of the RL becomes evident around the 220th round, indicating that the RL agent needs time to accumulate experiences and learn the optimal actions. In the final round of the simulation, the proposed RLDCSSA-CDG achieves a reduction in accumulated energy consumption of 8.93% compared to RLSSA-CDG and 38.8% compared to RLDCA for CDG. These results highlight the effectiveness of the RLDCSSA-CDG algorithm in enhancing energy efficiency in WSNs. By integrating sleep scheduling with reinforcement learning, the algorithm not only minimizes energy consumption but also enhances the overall sustainability of the network.

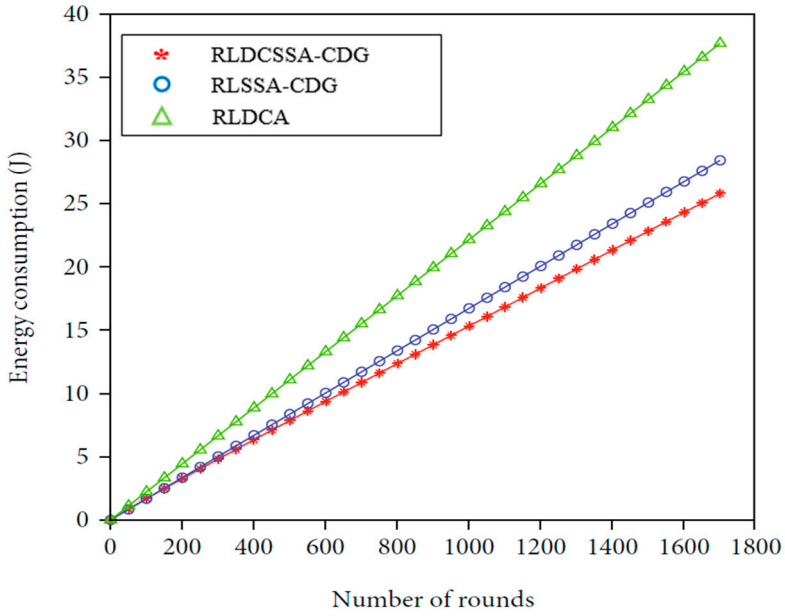

Figure 10. Energy consumption scenarios for RLDCSSA-CDG, RLSSA, and RLDCA.

5.2. Total Data Transmission

Each packet sent from the source to the destination includes a parameter known as time-to-live (TTL). The TTL is decreased by 1 each time a packet is transmitted from a sender to a receiver, regardless of whether the transmission is successful. If the TTL reaches 0 before the packet reaches its destination, the delivery is considered a failure. Figure 11 illustrates the total data transmissions in the wireless sensor network (WSN) for the three algorithms: RLDCSSA-CDG, RLSSA-CDG, and RLDCA for CDG. The results indicate that the proposed RLDCSSA-CDG achieves fewer total data transmissions than the other two algorithms. By the end of the 500th round, RLDCSSA-CDG reduced total data transmissions by 22.7% compared to RLSSA-CDG and 63.3% compared to RLDCA for CDG. This notable reduction in total data transmissions is even more significant than the decrease in energy consumption. This disparity can be attributed to two factors: consumption in WSNs is closely related to the distance between nodes, which can increase energy usage, and the RLDCSSA-CDG algorithm employs an on-demand transmission strategy. This means that data transmission occurs only after the sender establishes communication with the receiver, leading to a higher likelihood of successful packet delivery. The results confirm the advantages of the RLDCSSA-CDG algorithm, particularly in reducing total data transmissions while maintaining efficient energy consumption. By optimizing the transmission process, the algorithm enhances the overall performance of the WSN.

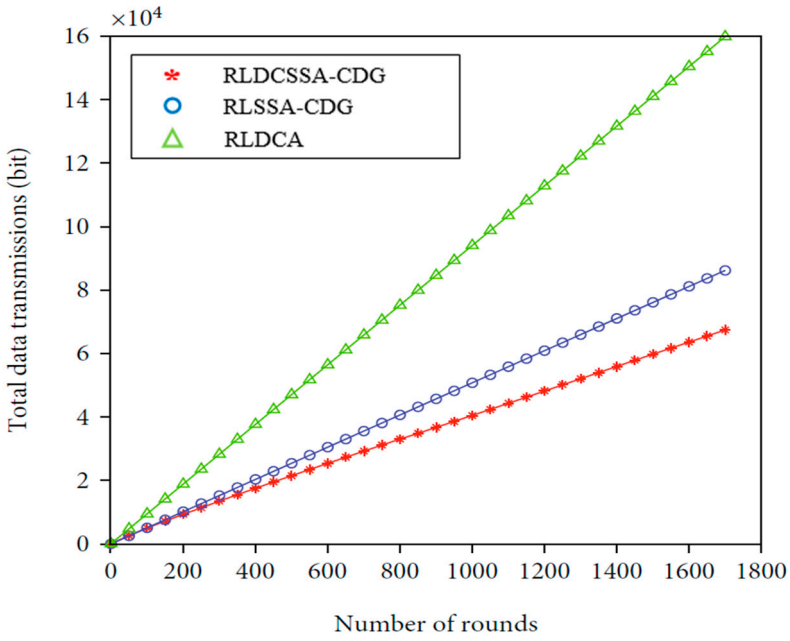

Figure 11. Total data transmission scenarios for RLDCSSA-CDG, RLSSA, and RLDCA.

5.3. Data Recovery Accuracy

The sink reconstructs the original data X from the received CS measurements. The mean square error (MSE) is used to evaluate the data recovery accuracy. It is calculated using Equation (13):

$$MSE = \frac{\|\hat{X} - X\|_2^2}{\|X\|_2^2} \tag{13}$$

where $\hat{X}$ is the recovery of the original data. If the MSE of the recovered data is no more than 10^{-2}, data reconstruction is deemed successful. The accumulated MSE after 100 rounds indicates that RLDCSSA-CDG has a lower MSE than the other two algorithms as shown in Figure 12, with average values of $MSE_{RLDCSSA\text{-}CDG}$ = 0.0005, MSE_{RLDCA} = 0.001, and MSE_{RLSSA} = 0.002. This indicates a 91.1% improvement in data recovery accuracy for RLDCSSA-CDG compared to the other algorithms.

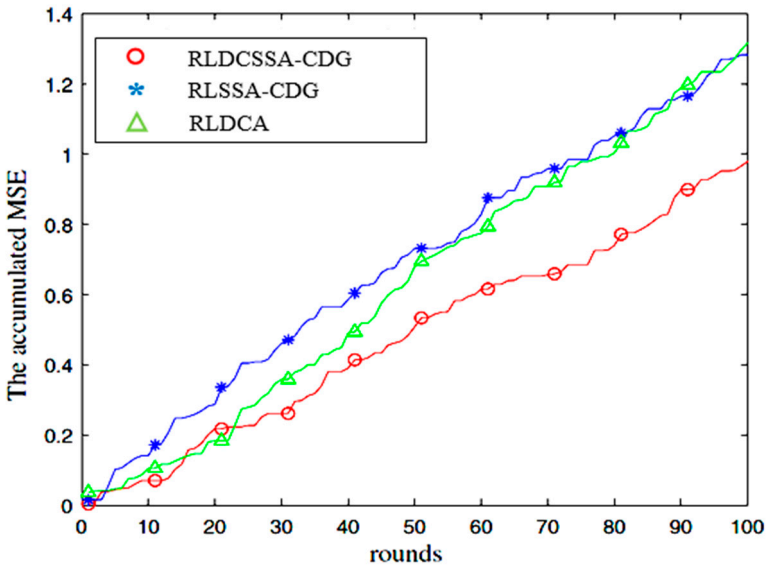

Figure 12. Data recovery accuracy scenarios for RLDCSSA-CDG, RLSSA, and RLDCA.

5.4. Reinforcement Learning Rewards

The average reward received by the agent in the RLDCSSA-CDG algorithm increases steadily with each learning round, indicating that the algorithm is convergent. This convergence suggests that the agent effectively learns from its experiences over time, optimizing its decision-making process. The parameter c in Equation (8) plays a crucial role in controlling the degree of exploration within the reinforcement learning framework. It determines how much the agent prioritizes exploring new actions versus exploiting known actions. Figure 13 illustrates the relationship between different values of c and the average reward achieved by the agent:

- When c = 1, the agent attains the maximum average reward, indicating that this value optimally balances exploration and exploitation.
- Values of c that are either lower or higher than 1 may lead to suboptimal performance, either by under-exploring or over-exploring.

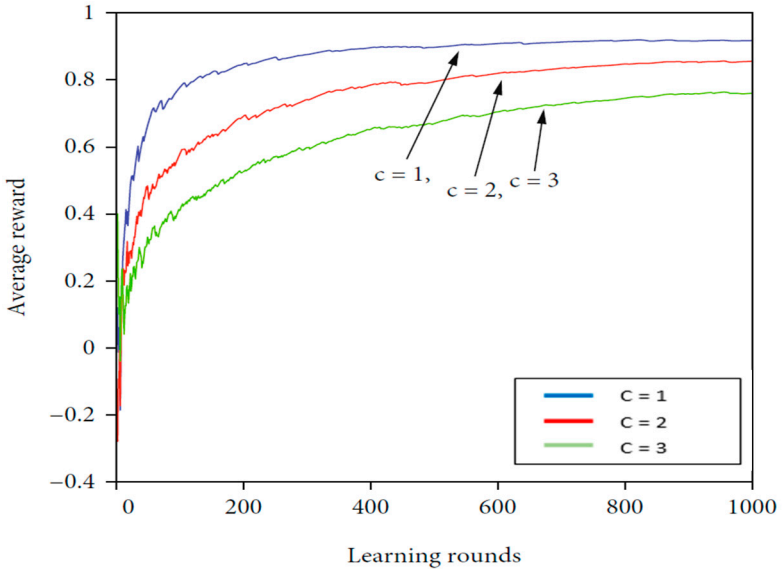

Figure 13. An average reward of node S_5 with different values of the parameter *c*.

Based on these observations, careful tuning of the exploration parameter c is essential for enhancing the performance of the RLDCSSA-CDG algorithm in managing data transmission and energy consumption in wireless sensor networks.

5.5. Average Delivery Latency

In Figure 14, with the increase in network scale, the average delivery latency rises. This is because when the network scale increases, based on the routing approach used in this simulation, on average, each packet has to be transmitted with more steps to its destination. Thus, the average delivery latency will undoubtedly increase. Specifically, it can be found that RLDCA achieves the highest latency (80 ms in the 100-node network, 185 ms in the 250-node network, and 302 ms in the 400-node network). This is because in RLDCA, whenever a sender intends to transmit a packet, it has to request their neighbors' information from the receiver and wait for the response from the receiver. This process will continue until the packet reaches the destination or the TTL of the packet reaches 0, so each packet will suffer a large latency during the transmission process. RLSSA and RLDCSSA-CDG achieve nearly the same latency, while the latency in RLDCA is slightly about 15%, which is lower than that in RLSSA and RLDCSSA-CDG. This can be explained by the fact that both RLSSA and RLDCSSA-CDG do not require nodes to periodically exchange information, so the time used for periodic information exchange can be saved. However, in

RLSSA, nodes have to request their neighbors' information for future wake-up prediction. The proposed RLDCSSA-CDG achieves the lowest latency (95 ms in the 100-node network, 110 ms in the 250-node network, and 165 ms in the 400-node network). Therefore, the proposed method improves the latency because it does not require periodic information exchange, and unlike RLDCA, it also does not require nodes to request their neighbors' information for prediction, so the corresponding time is saved.

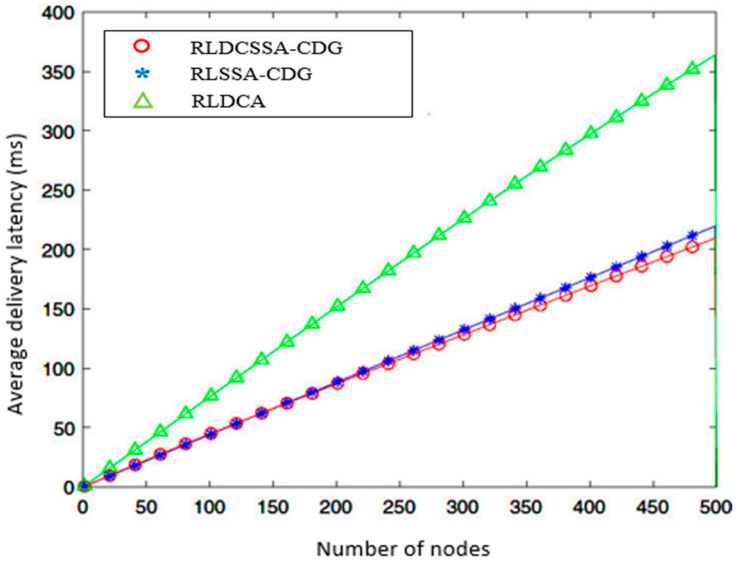

Figure 14. Average delivery scenarios for RLDCSSA-CDG, RLSSA, and RLDCA.

5.6. Network Lifetime Longevity

While minimizing energy consumption is crucial, it does not necessarily equate to maximizing the longevity of WSNs. The lifetime of a WSN is defined by its continuous operational duration, and when a node runs out of energy, it is considered to have "died." The FND metric is commonly used to assess the lifetime of WSNs, indicating when the first node exhausts its energy. This metric reflects the overall health of the network and its resilience. A well-balanced energy consumption among nodes is vital to prevent premature energy depletion in several nodes, which can significantly affect the FND. This load balancing is a key consideration in the design of the RLDCSSA-CDG algorithm. The simulation results demonstrate the effectiveness of the RLDCSSA-CDG in prolonging the lifespan of WSNs: $FND_{RLDCSSA\text{-}CDG} = 451$, $FND_{RLSSA} = 287$, and $FND_{RLDCA} = 61$. These results indicate that the rate at which nodes deplete their energy is significantly slower in the RLDCSSA-CDG compared to the other algorithms. The proposed RLDCSSA-CDG algorithm extends the WSN's lifetime by 77.3% relative to the compared algorithms, as illustrated in Figure 15. This longevity is attributed to the algorithm's ability to balance energy consumption effectively, thereby maintaining more nodes in operational status for a longer duration. Overall, the RLDCSSA-CDG algorithm not only emphasizes energy efficiency but also significantly enhances the operational lifespan of WSNs. By focusing on load balancing and effective energy management, the algorithm ensures a more sustainable and resilient network.

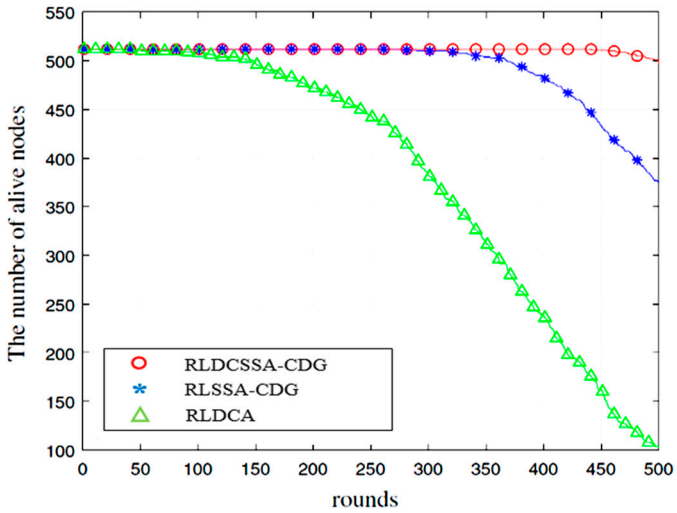

Figure 15. Network lifetime longevity scenarios for RLDCSSA-CDG, RLSSA, and RLDCA.

6. Conclusions

This paper introduced a reinforcement learning-based dynamic clustering of sleep scheduling algorithm (RLDCSSA-CDG) for compressive data gathering in WSNs. The carefully designed RL algorithm guides these agents in dynamically selecting clusters with strong data correlation and appropriate distances. This dynamic scheme is well suited for adapting to time-varying data within the network. Additionally, the paper presented an adaptive sleep scheduling algorithm to manage the wake-up process of forwarder nodes, allowing them to wake up just at the "right time" to receive incoming packets, forward them, and quickly return to sleep mode. The selection of active nodes was modeled as a Markov decision process (MDP), and a Q-learning algorithm was employed to find the optimal decision strategy. Unlike traditional duty cycling, this algorithm divides the time axis into several time slots, allowing each node to autonomously decide when to sleep, listen, or transmit, thereby eliminating the need for direct communication. Simulation results validate the effectiveness of the proposed algorithm by comparing it with the RLSSA and RLDCA algorithms. The proposed RLDCSSA-CDG reduces total data transmissions by 22.7% and 63.3% and energy consumption by 8.93% and 38.8%, respectively, and the proposed method achieves the lowest latency compared to the two contrastive algorithms. Furthermore, the proposed algorithm increases the whole network lifetime by 77.3% and promotes data recovery accuracy by 91.1% relative to the compared algorithms. As a result, the proposed RLDCSSA-CDG is a lightweight algorithm that can be implemented in resource-constrained WSNs. Additionally, due to clustering and distributed learning properties, the proposed RLDCSSA-CDG may be simply expanded to large-scale WSNs.

Author Contributions: Conceptualization, A.N.E.-S. and I.F.M.; methodology, A.N.E.-S.; software, A.N.E.-S.; validation, A.N.E.-S., I.F.M. and E.H.A.; formal analysis, A.N.E.-S.; investigation, A.N.E.-S.; resources, A.N.E.-S.; data curation, A.N.E.-S.; writing—original draft preparation, A.N.E.-S. and I.F.M.; writing—review and editing, A.N.E.-S., I.F.M. and M.A.M.; visualization, A.N.E.-S., I.F.M. and M.A.M.; supervision, I.F.M. and M.A.M.; project administration, A.N.E.-S. and I.F.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data may be available upon request.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Saleh, S.A.S.; Mabrouk, T.F.; Tarabishi, R.A. An improved energy-efficient head election protocol for clustering techniques of wireless sensor network (June 2020). *Egypt. Inform. J.* **2021**, *22*, 439–445. [CrossRef]
2. Negra, R.; Jemili, I.; Belghith, A.J.P.C.S. Wireless body area networks: Applications and technologies. *Procedia Comput. Sci.* **2016**, *83*, 1274–1281. [CrossRef]
3. Zahhad, M.A.; Farrag, M.; Ali, A. A Comparative Study of Energy Consumption Sources for Wireless Sensor Networks. *Int. J. Grid Distrib. Comput.* **2015**, *8*, 65–76. [CrossRef]
4. Luo, C.; Wu, F.; Sun, J.; Chen, C.W. Compressive data gathering for large-scale wireless sensor networks. In Proceedings of the 15th Annual International Conference on Mobile Computing and Networking, Beijing, China, 20–25 September 2009; pp. 145–156.
5. Prabha, M.; Darly, S.S.; Rabi, B.J. A novel approach of hierarchical compressive sensing in wireless sensor network using block tri-diagonal matrix clustering. *Int. J. Grid Distrib. Comput.* **2021**, *168*, 54–64.
6. Xiang, L.; Luo, J.; Rosenberg, C. Compressed data aggregation: Energy-efficient and high-fidelity data collection. *IEEE/ACM Trans. Netw.* **2012**, *21*, 1722–1735. [CrossRef]
7. Abdulzahra, A.M.K.; Al-Qurabat, A.K.M.; Abdulzahra, S.A. Optimizing energy consumption in WSN-based IoT using unequal clustering and sleep scheduling methods. *Internet Things* **2023**, *22*, 100765. [CrossRef]
8. Lin, C.-C.; Peng, Y.-C.; Chang, L.-W.; Chen, Z.-Y. Joint deployment and sleep scheduling of the Internet of things. *Wirel. Netw.* **2022**, *28*, 2471–2483. [CrossRef]
9. Yin, J.; Yang, Y.; Wang, L. An Adaptive Data Gathering Scheme for Multi-Hop Wireless Sensor Networks Based on Compressed Sensing and Network Coding. *Sensors* **2016**, *16*, 462. [CrossRef] [PubMed]
10. Chen, W.; Wassell, I.J. Optimized node selection for compressive sleeping wireless sensor networks. *IEEE Trans. Veh. Technol.* **2015**, *65*, 827–836. [CrossRef]
11. Chen, W.; Wassell, I.J. Cost-aware activity scheduling for compressive sleeping wireless sensor networks. *IEEE Trans. Signal Process.* **2016**, *64*, 2314–2323. [CrossRef]
12. Aziz, A.; Singh, K.; Osamy, W.; Khedr, A.M. Effective algorithm for optimizing compressive sensing in IoT and periodic monitoring applications. *J. Netw. Comput. Appl.* **2019**, *126*, 12–28. [CrossRef]
13. Al-Tous, H.; Barhumi, I. Reinforcement learning framework for delay sensitive energy harvesting wireless sensor networks. *IEEE Sens. J.* **2020**, *21*, 7103–7113. [CrossRef]
14. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
15. Al Mazaideh, M.; Levendovszky, J. A multi-hop routing algorithm for WSNs based on compressive sensing and multiple objective genetic algorithm. *J. Commun. Netw.* **2021**, *23*, 138–147. [CrossRef]
16. Aziz, A.; Osamy, W.; Khedr, A.M.; El-Sawy, A.A.; Singh, K. Grey Wolf based compressive sensing scheme for data gathering in IoT based heterogeneous WSNs. *Wirel. Netw.* **2020**, *26*, 3395–3418. [CrossRef]
17. Mhatre, K.P.; Khot, U.P. Energy efficient opportunistic routing with sleep scheduling in wireless sensor networks. *Wirel. Pers. Commun.* **2020**, *112*, 1243–1263. [CrossRef]
18. Shagari, N.M.; Idris, M.Y.I.; Bin Salleh, R.; Ahmedy, I.; Murtaza, G.; Shehadeh, H.A. Heterogeneous energy and traffic aware sleep-awake cluster-based routing protocol for wireless sensor network. *IEEE Access* **2020**, *8*, 12232–12252. [CrossRef]
19. Rawat, P.; Chauhan, S. Particle swarm optimization based sleep scheduling and clustering protocol in wireless sensor network. *Peer Peer Netw. Appl.* **2022**, *15*, 1417–1436. [CrossRef]
20. Wang, X.; Chen, H.; Li, S. A reinforcement learning-based sleep scheduling algorithm for compressive data gathering in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2023**, *2023*, 28. [CrossRef]
21. Thein, M.C.M.; Thein, T. An energy efficient cluster-head selection for wireless sensor networks. In Proceedings of the 2010 International Conference on Intelligent Systems, Modelling and Simulation, Liverpool, UK, 27–29 January 2010; pp. 287–291.
22. Lin, C.; Han, G.; Qi, X.; Du, J.; Xu, T.; Martinez-Garcia, M. Energy-optimal data collection for unmanned aerial vehicle-aided industrial wireless sensor network-based agricultural monitoring system: A clustering compressed sampling approach. *IEEE Trans. Ind. Inform.* **2020**, *17*, 4411–4420. [CrossRef]
23. Aziz, A.; Singh, K.; Osamy, W.; Khedr, A.M. An efficient compressive sensing routing scheme for internet of things based wireless sensor networks. *Wirel. Pers. Commun.* **2020**, *114*, 1905–1925. [CrossRef]
24. Osamy, W.; Khedr, A.M.; Aziz, A.; El-Sawy, A.A. Cluster-tree routing based entropy scheme for data gathering in wireless sensor networks. *IEEE Access* **2018**, *6*, 77372–77387. [CrossRef]
25. Manchanda, R.; Sharma, K. Energy efficient compression sensing-based clustering framework for IoT-based heterogeneous WSN. *Telecommun. Syst.* **2020**, *74*, 311–330. [CrossRef]
26. Qiao, J.; Zhang, X. Compressive data gathering based on even clustering for wireless sensor networks. *IEEE Access* **2018**, *6*, 24391–24410. [CrossRef]

27. Wang, Q.; Lin, D.; Yang, P.; Zhang, Z. An energy-efficient compressive sensing-based clustering routing protocol for WSNs. *IEEE Sens. J.* **2019**, *19*, 3950–3960. [CrossRef]
28. Wang, X.; Chen, H.; Barcelo-Ordinas, J.M. A Reinforcement Learning-Based Dynamic Clustering Algorithm for Compressive Data Gathering in Wireless Sensor Networks. *Mob. Inf. Syst.* **2022**, *2022*, 2736734. [CrossRef]
29. Bai, T.; Yuan, S.; Li, X.; Yin, X.; Zhou, J. Multi-density clustering based hierarchical path planning. In Proceedings of the 2019 2nd International Conference on Artificial Intelligence and Big Data (ICAIBD), Chengdu, China, 25–28 May 2019; pp. 176–182.
30. Komuraiah, B.; Anuradha, D. Efficient data gathering model with energy based routing for compressive sensing in multi-hop heterogeneous wireless sensor networks. *J. Theor. Appl. Inf. Technol.* **2024**, *102*, 5454–5468.
31. Batra, P.K.; Kant, K. LEACH-MAC: A new cluster head selection algorithm for Wireless Sensor Networks. *Wirel. Netw.* **2016**, *22*, 49–60. [CrossRef]
32. Moussa, N.; Nurellari, E.; El Belrhiti El Alaoui, A. A novel energy-efficient and reliable ACO-based routing protocol for WSN-enabled forest fires detection. *J. Ambient. Intell. Humaniz. Comput.* **2023**, *14*, 11639–11655. [CrossRef]
33. Heinzelman, W.B.; Chandrakasan, A.P.; Balakrishnan, H. An application-specific protocol architecture for wireless microsensor networks. *IEEE Trans. Wirel. Commun.* **2002**, *1*, 660–670. [CrossRef]
34. Altman, E. *Constrained Markov Decision Processes*; Routledge: London, UK, 2021.
35. Meyn, S. The Projected Bellman Equation in Reinforcement Learning. *IEEE Trans. Autom. Control* **2024**, *69*, 8323–8337. [CrossRef]
36. Skaltsis, G.M.; Shin, H.-S.; Tsourdos, A. A survey of task allocation techniques in MAS. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021; pp. 488–497.
37. Bowling, M.; Veloso, M. Multiagent learning using a variable learning rate. *Artif. Intell.* **2002**, *136*, 215–250. [CrossRef]
38. Rodrigues Gomes, E.; Kowalczyk, R. Dynamic analysis of multiagent Q-learning with ε-greedy exploration. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 369–376.
39. Ozdemir, S.; Xiao, Y. Secure data aggregation in wireless sensor networks: A comprehensive overview. *Comput. Netw.* **2009**, *53*, 2022–2037. [CrossRef]