Chapter 6

MS-based network cut detection and recovery scheme for IoT-enabled WSNs

6.1 Introduction

Wireless Sensor Networks (WSNs) are a major element of any Internet of Things (IoT) based system [138]. Nowadays, the IoT is used in a variety of applications, such as home automation, environmental monitoring, industry 4.0, smart city, and military surveillance [139, 140, 141, 142, 143]. In these applications, WSNs are used to monitor the environment and collect data from the monitoring environment. WSNs consist of low-priced, energy-constrained small sensor nodes. These sensor nodes communicate with each other to form an interconnected network. In WSNs, sensor nodes are prone to various faults due to energy depletion, software issues, and hardware malfunctions. On the other hand, sensor nodes are also damaged by environmental issues such as heavy rain, fire, and earthquakes. Faulty nodes disrupt network operations and degrade the performance of the network. Failure of some nodes prevents the active node data transmission. It creates network cuts and divides the network into disconnected/isolated

segments. These isolated segments are unable to transmit their data to the Base Station (BS)/sink due to the unavailability of effective data routing paths.

Some existing approaches reestablish the connection between disconnected segments by placing relay nodes between them [77, 23, 78]. However, it incurs a large amount of additional hardware cost. Furthermore, nodes near the sink relay a vast amount of data as compared to the faraway nodes from the sink. Therefore, the energy consumption rate of the sink nearest region nodes is very high as compared to the faraway nodes. As a result, the sink nearest region nodes die early and also create network cuts. This problem is known as the energy hole problem, which decreases the overall network performance [144]. Recently, a Mobile Data Collector (MDC) based data routing mechanism is very popular to avoid energy hole problems. In these approaches, some Rendezvous Points (RPs) are selected throughout the deployment area. An MDC periodically visits the RPs to collect data from the deployed sensor nodes. Sensor nodes send data to MDC with fewer hops and reduce energy usage of sensor nodes. However, existing MDC based approaches are unable to detect network cuts that occur within the network lifetime. Additionally, they are also unable to collect data from isolated segments that are created due to network cuts [145, 21].

Recently, some researchers have proposed MDC-based data collection techniques for predefined isolated network segments [80, 84, 146]. However, these approaches do not address the identification of network cuts and the formation of isolated segments within WSN. Therefore, the existing approaches are unable to detect the partition and also unable to collect data from the isolated segments. Furthermore, these approaches did not consider the energy consumption of sensor nodes while designing the data collection path [146, 15]. Most of these approaches select a sensor node as an RP that is close to the boundary of a predefined isolated segment. It causes uneven energy consumption at sensor nodes that lead to the premature death of the network. It also drastically reduces network performance.

Therefore, this chapter proposes an MDC-based novel network cut detection algorithm. The proposed network cut detection algorithm is able to detect network cuts that occur due to a single Articulation Point Node (APN) failure as well as multiple non-APN failures within the network lifetime. This chapter also proposes a novel network recovery algorithm that identifies dead nodes and recovery nodes. Furthermore, a Reinforcement Learning Brain Storm Optimization (RLBSO) algorithm based approach is proposed for optimal number of RPs selection and optimal data collection path design for MDC. The RLBSO algorithm has a higher success rate and performs better in exploration and exploitation stages as compared to classic meta-heuristic approaches [147]. The simulation results indicate that the proposed approach outperforms existing state-of-the-art approaches in terms of network lifetime, energy consumption, data delivery ratio, and latency. The following are the main contributions of this chapter.

- This chapter proposes an MDC-based novel network cut detection algorithm and a network recovery algorithm. The proposed algorithm identifies the occurrence of network cuts and the formation of isolated segments. Furthermore, if any network cut is detected by the proposed cut detection algorithm, then the proposed network recovery algorithm ensures the collection of data from all isolated segments with minimum delay.

- This chapter also proposes an RLBSO algorithm-based data collection approach for WSNs. The proposed approach selects the optimal RPs and designs the optimal data collection path for MDC movement.

- Extensive simulations and testbed experiments have been performed that show that the proposed approach outperforms the existing state-of-the-art approaches.

Table 6.1: Terminologies and definitions

Terminologies	***Definition***
$\mathcal{N}$	Number of sensor nodes in the network.
S	Set of sensor nodes. $S = s_1, s_2, s_3, s_4.....s_{\mathcal{N}}$.
$\mathcal{M}$	Number of RPs.
$\mathbb{R}$	Rendezvous points set. $\mathbb{R} = \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3....\mathcal{R}_{\mathcal{M}}$.
rc	Sensor nodes' communication range.
$\hat{\mathrm{D}}(x, y)$	Euclidean distance between node x and node y.
ξ_i	Remaining energy of node i.
$\mathcal{W}_{\rangle}$	Weight function.
$\mathbb{AJ}$	Adjacency list of nodes in the network.
$\mathbb{AP}$	List of APNs.
$\mathbb{F}$	List of faulty nodes.
$\mathbb{D}$	List of dead nodes.
$\mathbb{RL}$	List of recovery nodes.
$\mathbb{P}$	MDC path.

6.2 Energy and Network Model

In this chapter, a first-order radio energy model is used to compute the energy consumption of the sensor nodes. It is similar to [62]. Each sensor node consumes $\mathcal{E}_{tx}(l, d)$ amount of energy for transmitting l bit of data over the distance d. Sensor nodes also consume $\mathcal{E}_{rx}(l)$ amount of energy for receiving l bit data packet. The electronic circuit consumes $\mathcal{E}_{elec}$ energy. The $\mathcal{E}_{tx}(l, d)$ and $\mathcal{E}_{rx}(l)$ are calculated by the following equations.

$$\mathcal{E}_{(tx)}(l, d) = \begin{cases} l \times \mathcal{E}_{elec} + l \times e_{fs} \times d^2, & d < d_o \\ l \times \mathcal{E}_{elec} + l \times e_{mp} \times d^4, & d \geq d_o \end{cases} \tag{6.1}$$

$$\mathcal{E}_{rx}(l) = \mathcal{E}_{\text{elec}} \times l. \tag{6.2}$$

where e_{fs} is amplifier energy for the free space and e_{mp} is the multi-path fading channel energy. The d_o is threshold distance which is calculated as $\sqrt{\frac{e_{fs}}{e_{mp}}}$.

In this chapter, $\mathcal{N}$ number of sensor nodes are deployed in the $\mathcal{D}$ X $\mathcal{D}$ $[m^2]$ monitoring area. The network is represented as graph $\mathcal{G}$= (S_i, E_i), where sensor nodes are represented vertex S, and the connection between them is represented as an edge E. The sensor nodes $S = s_1, s_2, .., s_{\mathcal{N}}$ are randomly distributed in the monitoring area. The communication range of sensor nodes is rc. Two nodes have a data transmission link

between them only if $\hat{\mathrm{D}}(s_i, s_j) \leq rc$. The deployed sensor nodes are static, and an MDC moves in the network at a constant speed. MDC has adequate battery, processing, and storage space. The sensor nodes have limited energy and storage capacity. The terminologies used in this chapter are described in Table 6.1.

6.3 Problem Statement

This section defines the problem of designing a data collection tour for MDC in WSNs. Additionally, if any network cut is generated within the network, that must be detected and included within the data collection tour of the MDC. The objective is to minimize the energy consumption of SNs and reduce the MDC tour length. Given a set of sensor nodes in a WSN, the problem is to select an optimal set of RPs and design an optimal path for MDC in such a way that it meets the defined objectives. The mathematical presentation of the problem is as follows.

$$\text{Minimize } Obj = |\mathbb{P}| \tag{6.3}$$

where

$$|\mathbb{P}| = \min \left\{ \sum_{i=1}^{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} \psi_{i,j} \hat{D} \left(\mathcal{R}_i, \mathcal{R}_j \right) \right\} \tag{6.4}$$

$$\psi_{i,j} = \begin{cases} 1 & \text{if RP}\mathcal{R}_i \text{ comes immediately before RP}\mathcal{R}_j \text{ in } |\mathbb{P}|, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

such that

$$\sum_{j=1}^{\mathcal{M}} x_{ij} \geq 1 \quad \forall i \in \{1, 2, \ldots, \mathcal{K}\} \tag{6.6}$$

$$\prod_{\mathcal{R}_i \in R} C \left(\mathcal{R}_i, \mathbb{P} \right) = 1, \tag{6.7}$$

$$\sum_{j=1}^{\mathcal{M}} \mathcal{A}(i, j) = 1, \text{for each } i = 1, 2, 3...\mathcal{N}. \tag{6.8}$$

where x_{ij}, C_i and $A_{i,j}$ are Boolean variables that are defined as follows.

$$x_{ij} = \begin{cases} 1 & \text{if RP } \mathcal{R}_j \text{ is present in segment } x_i \\ 0 & \text{otherwise.} \end{cases} \tag{6.9}$$

$$C_i = \begin{cases} 1 & \text{if RP } \mathcal{R}_i \text{ is visited only once in } |\mathbb{P}| \\ 0 & \text{otherwise.} \end{cases} \tag{6.10}$$

$$A_{i,j} = \begin{cases} 1 & \text{if } s_i \text{ is allotted to only RP } \mathcal{R}_j \\ 0 & \text{otherwise.} \end{cases} \tag{6.11}$$

The constraint (6.6) ensures that each segment contains at least one RP. In case there is no network cut, then $\mathcal{K} = 1$. The constraint (6.7) ascertain that all RPs are visited only once in the MDC tour. Furthermore, it also makes sure that all RPs are included in the tour. The constraint (6.8) ascertain that all sensor nodes are covered by an RP.

6.4 Proposed Work

In the proposed work, MDC moves in the network to collect data from sensor nodes. The proposed RLBSO algorithm is used to compute an optimal number of RPs and an MDC path for data gathering. After the RP selection, a set of Data Forwarder Nodes (DFNs) is selected. These DFNs act as cluster heads. DFNs collect data from sensor nodes and forward it to MDC. The MDC delivers that data to BS at the end of the tour. Furthermore, if any faulty nodes are detected during data collection, then MDC checks if there is a network cut or not using the proposed network cut detection algorithm. If any network cut is detected, then the proposed network recovery algorithm is executed to identify dead nodes ($\mathbb{D}$) and recovery nodes ($\mathbb{RL}$). MDC delivers the list of dead nodes and recovery nodes to BS. The BS identifies the isolated segments using the recovery

node list. The proposed RLBSO algorithm is applied to recalculate an optimal number of RPs in the network and design the optimal MDC path. The proposed RLBSO based optimal number of RP selection and path designing, DFN selection, network cut and recovery, and MDC based data gathering algorithms are described in following sections.

6.4.1 RLBSO based optimal number of RP selection and path designing mechanism

The proposed approach uses an RLBSO algorithm for the selection of an optimal number of RPs and an optimal data gathering path for MDC. The proposed approach selects energy-efficient RPs. RP is a data collection point where MDC halts and collects data from the DFNs. In the network, RPs are selected in such a way that they are close to the centre of the cluster. This decreases the average distance between sensor nodes and RP within the cluster, which further reduces the energy consumption of sensor nodes. Furthermore, the proposed approach also maximizes the in-degree of RPs. Having a high in-degree of RP will have more nodes within single-hop communication. This minimizes the energy expended in data aggregation and forwarding. The proposed approach also minimizes the distance of the selected RP from the centre of the network. This ensures a shorter travelling path for MDC. Furthermore, the proposed approach designs a travelling path for MDC that minimizes the total travelling distance of MDC. It significantly reduces data gathering delay. Fig. 6.1 shows the flowchart of the proposed RLBSO algorithm. The coordinates of the network centre (x_c, y_c) are calculated as follows.

$$x_c = \frac{1}{\mathcal{N}} \sum_{t=1}^{\mathcal{N}} x_i, y_c = \frac{1}{\mathcal{N}} \sum_{t=1}^{\mathcal{N}} y_i. \tag{6.12}$$

where $\mathcal{N}$ is the number of nodes in the WSNs. The distance of an RP $\mathcal{RP}_j$ from network centre (x_c, y_c) is calculated as follows.

$$\hat{\mathrm{D}}_{cj} = \sqrt{(x_{\mathcal{R}_j} - x_c)^2 + (y_{\mathcal{R}_j} - y_c)^2}. \tag{6.13}$$

The in-degree of an RP $\mathcal{R}_j$ is calculated as follows.

$$\mathcal{N}_{\mathcal{ND}}(\mathcal{R}_j) = \sum_{i=1}^{\mathcal{N}} V_i, \text{ where } \mathrm{V_i} = \begin{cases} 1, \text{ if } s_i \in S \& \hat{\mathrm{D}}(s_i, \mathcal{R}_j) \leq rc, \\ 0, \text{ otherwise.} \end{cases} \tag{6.14}$$

where rc is communication range of sensor nodes and $\hat{\mathrm{D}}(s_i, \mathcal{R}_j)$ is distance between sensor nodes and RP. The centre of a cluster is calculated as follows.

$$x_k = \frac{1}{SN_k} \sum_{i=1}^{SN_k} x_i, y_k = \frac{1}{SN_k} \sum_{i=1}^{SN_k} y_i. \tag{6.15}$$

where SN_k is the number of nodes in k^{th} cluster. The distance of an RP $\mathcal{R}_j$ from cluster centre (x_k, y_k) is calculated as follows.

$$\hat{\mathrm{D}}_{kj} = \sqrt{(x_{\mathcal{R}_j} - x_k)^2 + (y_{\mathcal{R}_j} - y_k)^2}. \tag{6.16}$$

The total travelling distance L for MDC is calculated as follows.

$$L = \sum_{i=1}^{\mathcal{M}} \sum_{j=1}^{\mathcal{M}} K_{ij} P_{ij}. \tag{6.17}$$

where $P_{ij} = \hat{\mathrm{D}}(\mathcal{R}_i, \mathcal{R}_j)$, and $K_{ij} = 1$ if link connecting $\mathcal{R}_i$ to $\mathcal{R}_j$ is included in the path.

Fitness function: The fitness function for the proposed approach is calculated as follows.

$$Fitness \ = Minimize\{w_1 * max(\hat{\mathrm{D}}_{cj}) + w_2 * max(\hat{\mathrm{D}}_{kj}) + w_3 * L - w_4 * min(\mathcal{N}_{\mathcal{ND}}(\mathcal{R}_j))\}. \tag{6.18}$$

Such that:

$$w_1 + w_2 + w_3 + w_4 = 1 \tag{6.19}$$

where w_1, w_2, w_3, and w_4 are the weights assigned to the parameters. The algorithm 6 shows the proposed RLBSO-based algorithm. RLBSO [147] optimizes RP selection

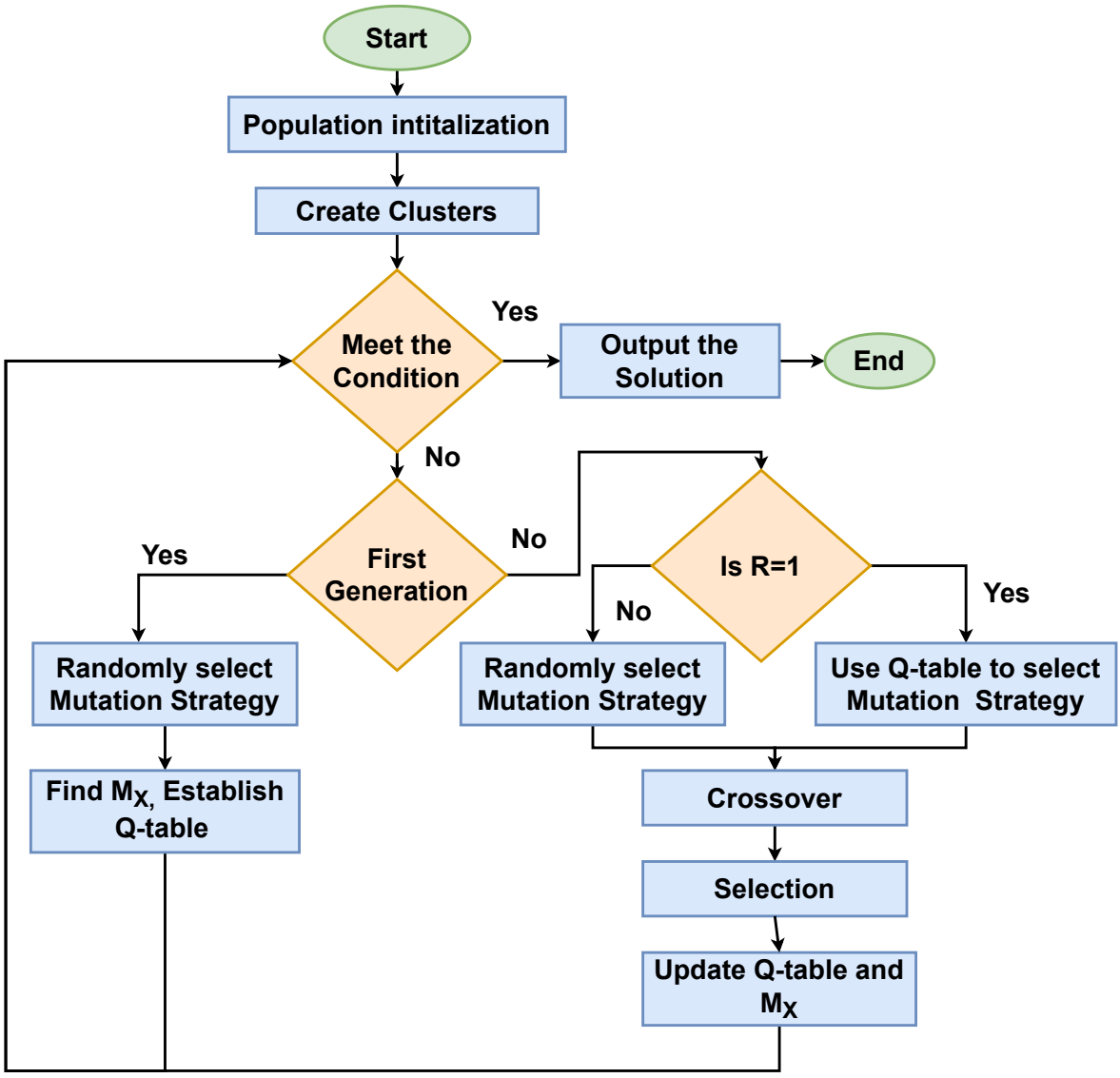


Figure 6.1: Proposed RLBSO approach.

and MDC path design by integrating reinforcement learning into the Brain Storm Optimization (BSO) algorithm. It consists of three major steps: clustering, generating new solutions, and selecting individuals. Initially, the solution space is randomly generated and divided into clusters. Cluster centres are replaced by random solutions with a probability of p_r. Four mutation strategies, such as elites, global best, cluster centres, and historical individuals, are used to update solutions. The control factor λ is determined

Algorithm 6: Proposed RLBSO-based Algorithm

Output: Parent of each SN, $\mathbb{P} = [\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n]$, Set of RPs RP

Initialize the population $x_i \epsilon \mathcal{X}$, $\forall i \leq 1 \leq \mathcal{P}$
Calculate the fitness value of each search agent using equation 6.18
while $t < T$ **do**
    Divide the $\mathcal{P}$ search agents into $\mathcal{K}$ clusters
    Select the best search agent in cluster j as the centre
    if $t < 1$ **then**
        Generate new search agents by randomly using four strategies
        Calculate the Q-learning table and get the individual M_x
    else
        for i=1 *to* $\mathcal{P}$ **do**
            Calculate the correlation coefficient $\mathcal{R}$
            if $\mathcal{R}$ =1 **then**
                Update search agents by using four strategies according to Q-table
            else
                Select four strategies randomly and update search agents
            Calculate the score of the search experiment of the current population
            Crossover operation
        Calculate the Q-learning table and get the individual M_x
    Calculate the fitness of each search agent by applying equation 6.18
    Find the best fit solution $\mathcal{P}_{best}$
$fitnessP_{best} = min(fitness)$
$globalminimum = fitnessP_{best}$
Return the best solution $\mathcal{F}itness\mathcal{P}_{best}$

by the following equation.

$$\lambda = e^{1-T/(T-t+1)} \tag{6.20}$$

where T is the total number of iterations, and t is the current iteration. Mutation strategies differ based on their use of elites, global best, cluster centres, and historical solutions, which are defined as follows.

$$X^t_{\text{selected },j} = X^t_{\text{best},j} + \lambda \left(X^t_{i_1,j} - X^t_{i_2,j} \right). \tag{6.21}$$

$$X^t_{\text{selected},j} = X^t_{i,j} + \lambda \left(X^t_{\text{gbest},j} - X^t_{i,j} \right) + \lambda \left(X^t_{p1,j} - X^t_{p2,j} \right). \tag{6.22}$$

$$X^t_{\text{selected},j} = X^t_{p\_\text{center},j} + \lambda \left(X^t_{p1,j} - X^t_{p2,j} \right). \tag{6.23}$$

$$X^t_{\text{selected},j} = X^t_{i,j} + F * \left(X^{t-1}_{i_1,j} - X^t_{i,j} \right) + F * \left(X^{t-1}_{i_2,j} - X^{t-1}_{i_3,j} \right). \tag{6.24}$$

where $X^t_{\text{best},j}$ represents the elites, and $X^t_{i_1,j}$ and $X^t_{i_2,j}$ are two different solutions belonging to different clusters. $X^t_{i,j}$ is the current solution. $X^t_{\text{gbest},j}$ is the global best. X_{p_center} is a cluster center, $X^t_{p1,j}$ and $X^t_{p2,j}$ are two different solutions from a cluster. $X^{t-1}_{i_1,j}$, $X^{t-1}_{i_2,j}$, and $X^{t-1}_{i_3,j}$ are solutions from historical population. F is a standard Gaussian distributed random number. Different mutation strategies are applied at different evolutionary stages. Q-learning is applied for the selection of optimal strategies in place of random selection. A Q-Table matrix is created, and the population is initialized. The population is updated with four mutation strategies. A score is computed based on the improvement, and the Q-table matrix is used to select a strategy for updating individuals from the second generation. The following equation is used to calculate the score.

$$\text{score}(t+1)^i = \text{score}(t)^i + \varepsilon. \tag{6.25}$$

$$\varepsilon = \frac{\sum_{n=1}^{mut^{l'}_n} |1|}{\sum_{n=1}^{mut^{i}_n} |1|}. \tag{6.26}$$

where mut^i_n is the count of members using mutation strategy and $mut^{l'}_n$ is the count of successful members. The individual with the best fitness value is selected as the optimal individual M_x. Each individual x_i in the population is evaluated for its correlation with M_x. If a linear correlation exists, the Q-table guides the mutation strategy selection. Otherwise, the individual randomly selects a strategy. Correlation coefficient $\rho(X, Y)$ is calculated as follows.

$$\begin{aligned} \rho_{(X,Y)} &= \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y} \\ &= \frac{E(XY) - E(X)E(Y)}{\sqrt{E\left(X^2\right) - E^2(X)}\sqrt{E\left(Y^2\right) - E^2(Y)}}. \end{aligned} \tag{6.27}$$

where $E(X)$ and $E(Y)$ are means of X and Y. Next, a crossover operation is ap-

plied, which modifies parts of the solution. It enhances rotation invariance and global optimization ability. The individuals become more correlated as the algorithm iterates. Re-initializing the cluster centre is insufficient to escape the local optimum. The crossover operation enhances the optimality on all dimensions. It is applied as follows.

$$x^t_{new,j} = \begin{cases} x^t_{new,j}, & \text{if rand} \leq CP \text{ or } j = j_{\text{rand}} \\ x^t_{i,j}, & \text{otherwise.} \end{cases} \tag{6.28}$$

where $x^t_{new,j}$ is the new search agent, $x^t_{i,j}$ is the current search agent j is random number between 1 to D. D is the dimensions of individuals. CP is calculated as follows.

$$CP = 0.9 - 0.2e^{1-\frac{T}{T-t+1}}. \tag{6.29}$$

where T is maximum no of iterations and t is current iteration.

Lemma 1: The time complexity of the proposed RLBSO-based optimal number of RP selection and optimal data collection path design algorithm is $O(TM^2)$.

Proof: In the proposed RLBSO-based optimal number of RP selection and optimal data collection path design algorithm, the population is initialized with the complexity $O(M)$. The fitness of the population is calculated with $O(M)$ complexity. Next, the population is divided into clusters with $O(M^2)$ complexity. The new individuals are generated with $O(M^2)$ complexity. These steps are repeated for T iterations. Therefore, the total time complexity of the RLBSO-based optimal number of RP selection and optimal data collection path design algorithm is $O(M)+O(TM)+O(TM^2)+O(TM^2)$ $\approx O(TM^2)$, where M is the population size.

6.4.2 Data forwarder node selection

In the proposed scheme, a set of DFNs are selected in each cluster which act as cluster heads. The function of DFNs is to collect data from the rest of the sensor nodes

within the cluster and transmit it to MDC when it arrives at their RP. DFNs are also responsible for creating $\mathbb{F}$ and sending it to MDC. DFNs are selected based on the weight function $\mathcal{W}_{\rangle}$. The $\mathcal{W}_{\rangle}$ is calculated based on the residual energy of sensor nodes and distance to the RP. The nodes having $\mathcal{W}_{\rangle}$ value higher than a predefined threshold are selected as DFNs. The distance between RP and DFN is minimized to reduce energy consumption in data transmission to MDC.

$$\mathcal{W}_i \propto \xi_i \tag{6.30}$$

$$\mathcal{W}_i \propto \frac{1}{\hat{D}(s_i, \mathcal{R}_i)} \tag{6.31}$$

Combining Eq. (6.30) and (6.31), the $\mathcal{W}_{\rangle}$ is calculated as follows.

$$\mathcal{W}_i = \mathcal{K} * \frac{\xi_i}{\hat{D}(s_i, \mathcal{R}_i)}. \tag{6.32}$$

where $\mathcal{K}$ is a constant.

6.4.3 Network cut detection and recovery algorithm

The BS is aware of the locations of sensor nodes and creates an adjacency list $\mathbb{AJ}$ of communication links between nodes. The Depth First Search (DFS) algorithm is used to identify the APNs in the network. APNs are the nodes that can divide the network into two or more segments if they become faulty. Nodes in the network are classified as APN or non-APN. In the DFS tree, leaf nodes are non-APNs. A node a in a DFS tree is considered an APN if any of the following two conditions are met.

1. The node a serves as the root of the DFS tree and has at least two child nodes.

2. The node a is not the root of the DFS tree. However, it contains a child node b, and b has a subtree rooted with b. Furthermore, there is no back edge to any of the ancestors in the DFS tree of a with nodes in the subtree of b.

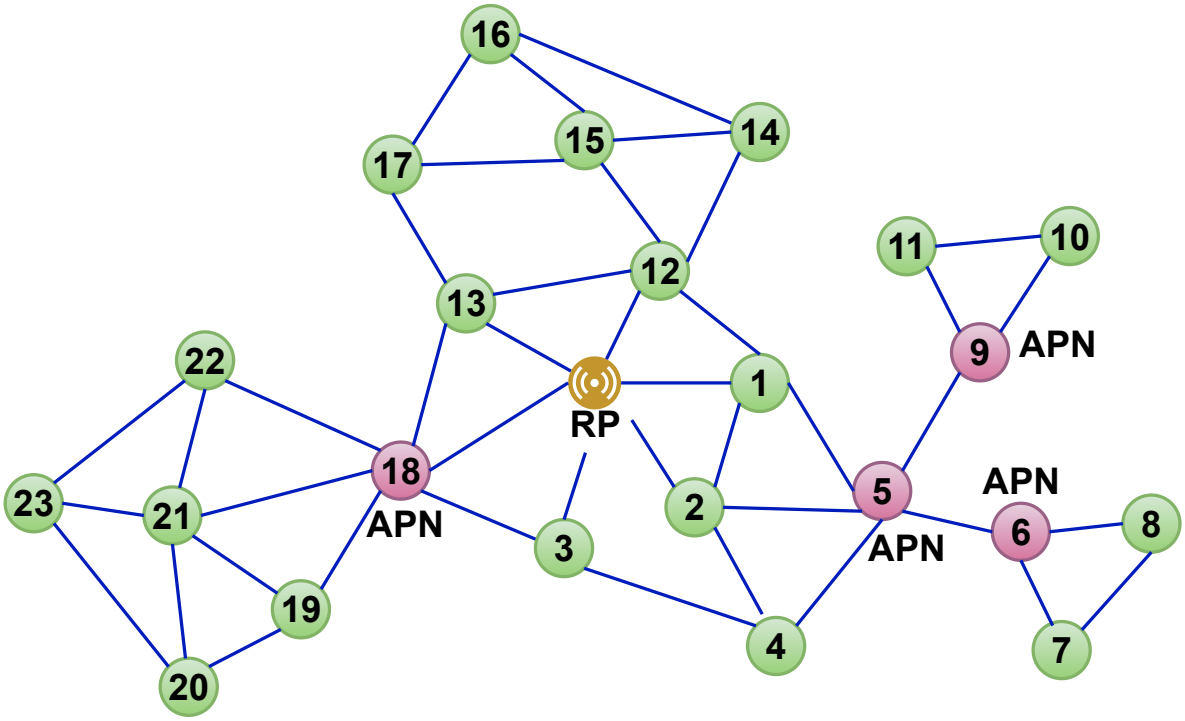


Figure 6.2: Example network.

The BS creates a list called $\mathbb{AP}$ that contains the IDs of the APNs. BS forwards the $\mathbb{AP}$ to the MDC. Sensor nodes ascertain their connectivity status by sending a periodic *Status_alive* message to their DFNs. If DFN does not receive *Status_alive* message from a sensor node within a predefined time period, then that sensor node is considered faulty. DFNs send the list of faulty nodes ($\mathbb{F}$) to MDC. The MDC detects cuts on the network based on the received list of faulty nodes. The following conditions are checked to detect a network cut.

1. If an APN becomes faulty, DFN does not receive *Status_alive* messages from other nodes that indicate a cut in the network. If $\mathbb{F}$ contains any APN, the recovery process is triggered. For instance, in Fig. 6.2, if APN 18 becomes faulty, DFN does not receive *Status_alive* messages from nodes 19, 20, 21, 22, and 23. It indicates a network cut.

2. If a single non-APN becomes faulty, then it is considered a single node failure. In

Algorithm 7: Detecting Network Cut

```
   Output: Network_cut_status
 1 S_i ← set of vertices in the i^th cluster
 2 PartitionNode = faulty nodes that are neighbours of non-faulty nodes or RP
 3 Initialize visited = φ
 4 RemVertices = vertices in S_i not in PartitionNode
 5 if RemVertices = φ then
 6     return false
 7 startNode = first vertex in RemVertices
 8 Initialize stack = φ
 9 stack.push(startNode)
10 Initialize VisitedCnt = 0
11 while stack ≠ φ do
12     currentNode ← stack.pop()
13     if not visited[currentNode] then
14         visited[currentNode] = true
15         visited[currentNode] = VisitedCnt + 1
16         neighbors = neighbors of currentNode in S_i
17         for each neighbor in neighbor do
18             if not visited[neighbor] and neighbor is not in PartitionNodes
                then
19                 stack.push(neighbor)
20 return VisitedCnt = size(RemVertices)
```

Fig. 6.2, if node 2 or node 16 fails, then it does not indicate any network cut.

3. If multiple non-APNs become faulty, then algorithm 7 is executed to check if there is a network cut or not. If a network cut is detected, then recovery is triggered. In Fig. 6.2, the failure of nodes 12 and 17 creates a network cut and recovery is needed. On the other hand, the failure of nodes 14 and 15 does not create a network cut.

Algorithm 7 is used to detect cuts in the network. Failure of multiple non-APNs can partition the network into multiple segments. A set $PartitionNodes$ is created for network cut detection. It contains the nodes from $\mathbb{F}$ that have a direct edge with non-faulty nodes or RP. DFS algorithm is executed to check the connectivity of the network after removing $PartitionNodes$. In the algorithm, the number of visited nodes

Algorithm 8: Network Recovery

```
Output: 𝔻, ℝ𝕃
Create groups of sensor nodes present in 𝔽 based on connectivity using 𝔸𝕁
Visit the group nearest to the RP
for each group g_i ∈ G do
    Add all nodes from g_i to unvisited
    while unvisited ≠ φ do
        if unvisited ∩ 𝔸ℙ ≠ φ then
            Visit the nearest APN
        else
            Visit the nearest node in unvisted
        Send Status_req message to nodes
        for each node s_j ∈ unvisited do
            if Status_alive received then
                Put node in ℝ𝕃
            else
                if Status_alive not received and (rc_j ≤ D̂(s_j, MDC)) then
                    Put node in 𝔻
            Remove node from 𝔽 and unvisited
return 𝔻 and ℝ𝕃
```

is counted. If $VisitedCnt$ is equal to the number of nodes in $RemVertices$, then there is no network cut in the network. If $VisitedCnt$ is less than the number of nodes in $RemVertices$, then a network cut is detected, and the recovery algorithm is triggered.

After detecting the network cut, the network recovery process is initiated. Algorithm 8 shows the proposed network recovery algorithm. First, sensor nodes present in $\mathbb{F}$ are divided into groups based on links between them. Next, MDC visits the nearest group of faulty nodes. MDC first visits the faulty APNs and then proceeds to visit non-APNs. If the $\mathbb{F}$ contains multiple APNs, then MDC visits them based on the nearest distance to their RP. It minimizes the distance travelled by MDC. After reaching a node, MDC broadcasts a status request *(St_req)* message. The nodes that receive *St_req* reply with *Status_alive* message. If *Status_alive* message is received from a node within a predefined time period, then MDC considers that node alive and removes it from $\mathbb{F}$. These nodes are then added to $\mathbb{RL}$. If MDC does not receive a reply from a node, then

that node is considered dead and added to $\mathbb{D}$. MDC delivers the $\mathbb{RL}$ and $\mathbb{D}$ to the BS at the end of the tour. BS executes the algorithm 6 to select RPs for new segments and design the data collection path for MDC.

***Lemma 2*:** The time complexity of the proposed network cut detection algorithm is $O(\mathcal{N} + E)$.

***Proof*:** In the proposed network cut detection algorithm, set of $RemVertices$ is created which takes $O(n)$ time, where n is the number of nodes in the cluster. Next, DFS traversal is applied to visit the nodes in $RemVertices$ to count the number of visited vertices. The complexity of DFS is $O(V + E)$, where V is the number of vertices and E is the set of edges. The total time complexity of the proposed network cut detection algorithm is $O(n) + O(V + E)$. Since $V = n$, the time complexity is $O(n + E)$. In the worst case scenario, $n = \mathcal{N}$. Thus, the time complexity of the proposed network cut detection algorithm is $O(\mathcal{N} + E)$.

***Lemma 3*:** The time complexity of the proposed network cut recovery algorithm is $O(\mathcal{N} + E)$.

***Proof*:** In the proposed network recovery algorithm, first sensor nodes are grouped based on $\mathbb{AJ}$, which takes $O(V + E)$ time. Variable V is the number of nodes, and E is the number of edges. In the worst-case scenario, $V = \mathcal{N}$. So time complexity of this step is $O(\mathcal{N} + E)$. In each group, MDC first visits APNs and then non-APN nodes. In the worst-case scenario, MDC visits every node, which takes $O(k \times n)$. Variable k is the number of groups, and n is the number of nodes per group. In the worst case, $k \times n = \mathcal{N}$. The total time complexity is $O(\mathcal{N} + E) + O(\mathcal{N})$. Hence, the total time complexity of the proposed network recovery algorithm is $O(\mathcal{N}+E)+O(\mathcal{N}) \approx (\mathcal{N}+E)$.

6.4.4 MDC-based data gathering approach

In the MDC-based data gathering phase, MDC moves in the network to collect data from sensor nodes. MDC visits RP and broadcasts an *announcement (ANN)* message

to the DFNs in the cluster. *ANN* message contains the id of the current RP R_{id}. DFNs send a *reply (REP)* message to the MDC. *REP* contains the id of the DFN and the current state of the buffer. MDC allocates a time slot to each DFN based on the received *REP* messages. Furthermore, DFNs transmit their data to MDC based on the received time slot. MDC sends a final check *(CHK)* message in the cluster to ensure all DFNs have sent their data to MDC. DFNs that receive *CHK* message check their buffer. If the buffer is empty, then that DFN has already transmitted its data to MDC. If the buffer is not empty, then the DFN transmits a *REP* message to MDC. MDC assigns a time slot to the DFN, and the DFN sends data to MDC. MDC goes to the next RP after collecting data from the current RP. After visiting every RP, MDC delivers the collected data to BS.

Lemma 4: The message complexity of the proposed MDC-based data gathering approach is $O(k\mathcal{M})$.

***Proof*:** In the proposed MDC-based data gathering approach, MDC reaches an RP and sends k *ANN* messages to DFNs. Next, k DFNs send *REP* message to the MDC. MDC assigns a time slot to each DFN. k DFNs transmit their data to MDC. MDC visits $\mathcal{M}$ RPs. Therefore, the total message complexity of the proposed MDC based data gathering approach is $O(k\mathcal{M} + k\mathcal{M} + k\mathcal{M} + k\mathcal{M}) \approx O(k\mathcal{M})$.

6.5 Performance Analysis

This section analyzes the performance of the proposed approach. The NS3 simulator is used for the simulation. The parameters used in the simulation are shown in Table 6.2. The performance of the proposed approach is compared with the Objective Variable Tour planning (OVTP) [82], GTSP-Based data collection Algorithm (GBA) [80], and Obstacle Aware Connectivity Restoration using Mobile Data carrier (OCRS-MD) [81] in terms of network lifetime, data collection ratio, energy consumption, and latency. Furthermore, network cut detection time and recovery time are also analyzed.

Table 6.2: Simulation parameters

Parameters	*Values*
WSN deployment area	1000 X 1000 m^2
Number of sensor nodes	200-600
Initial energy of each sensor node	1 J
$\mathcal{E}_{\text{elec}}$	50 nJ/bit
ε_{fs}	10 pJ/bit/m^2
ε_{mp}	0.0013 pJ/bit/m^4
Population size	40
Speed of MDC	2-8 m/s

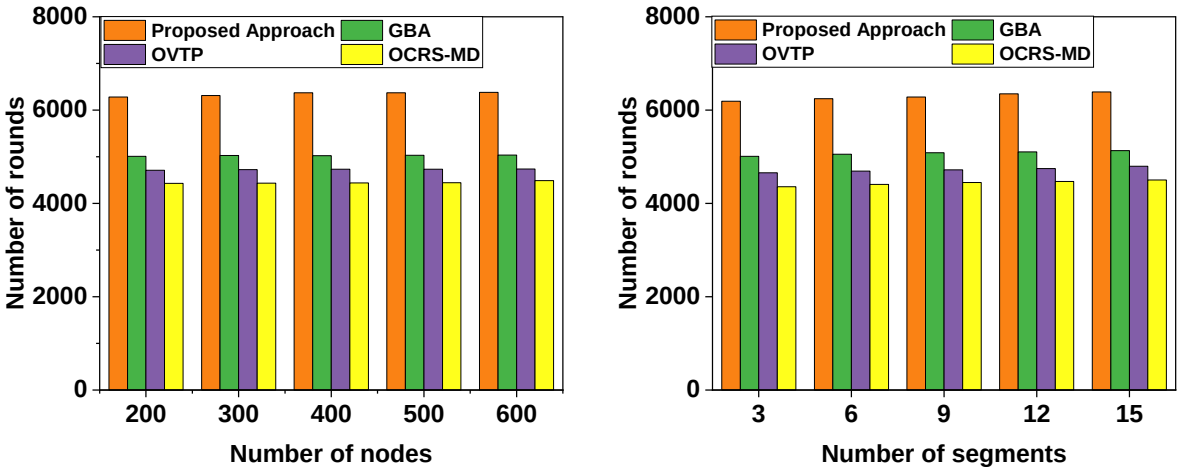


Figure 6.3: Network lifetime.

Fig. 6.3a depicts the network lifetime in different number of nodes. Fig. 6.3b shows the network lifetime in different number of network segments. The results show that the proposed approach improves the network lifetime by 42% than OCRS-MD, by 36 % than OVTP and by 27% than GBA approaches. The proposed approach selects a set of DFNs that forward sensor node data directly to MDC. It saves the energy of nodes. Furthermore, the proposed approach selects RP near the centre of the cluster, which reduces the energy consumption of sensor nodes. It results in a longer lifetime for sensor nodes, which further increases the overall network lifetime. Fig. 6.4a displays the total consumed energy in different number of nodes. Fig. 6.4b shows the total consumed energy of the network in different number of network segments. The results show that the proposed approach decreases the sensor nodes' energy consumption by 44%

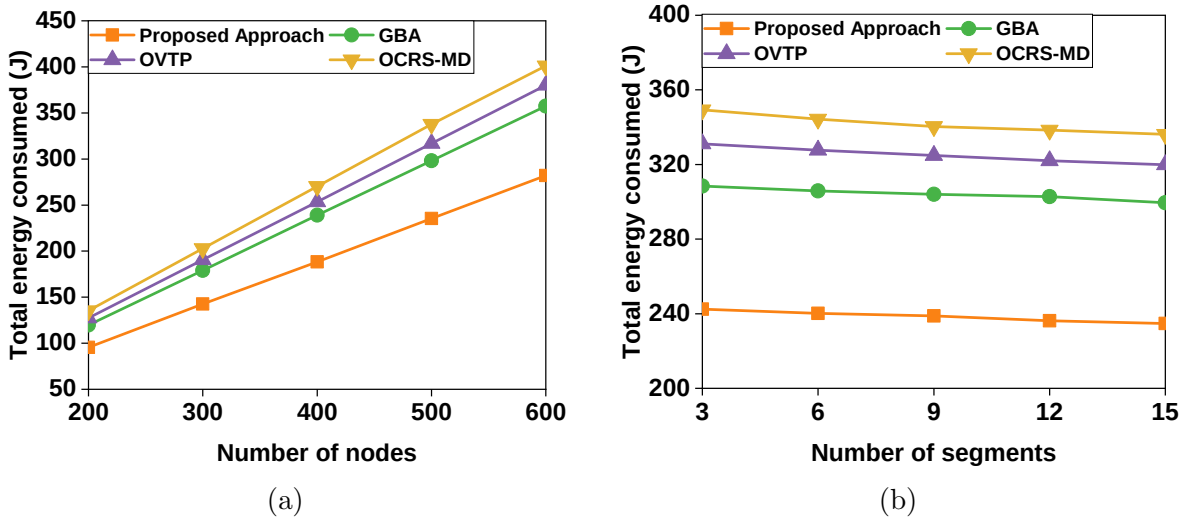


Figure 6.4: Energy consumed.

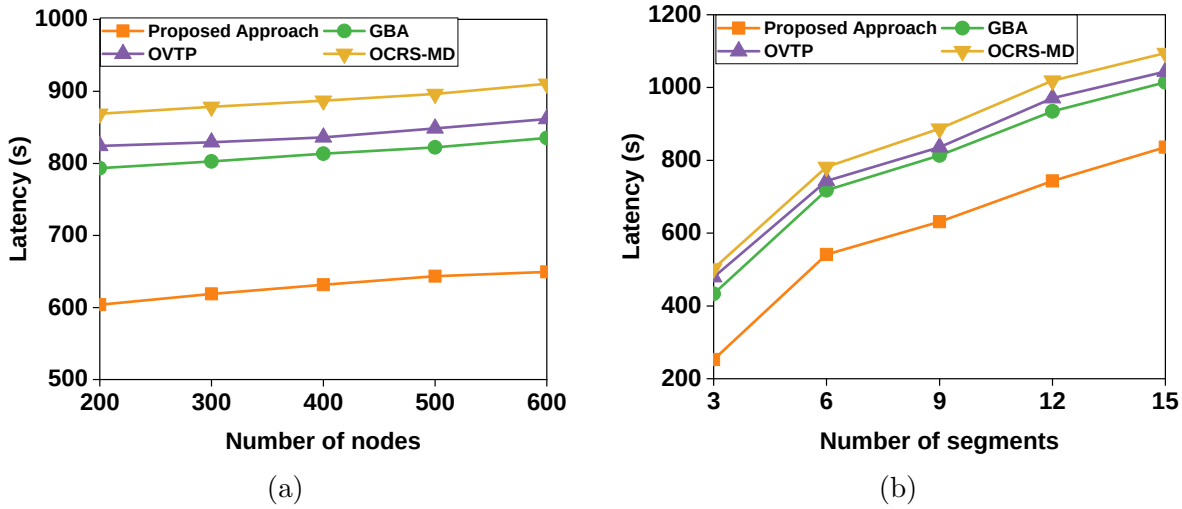
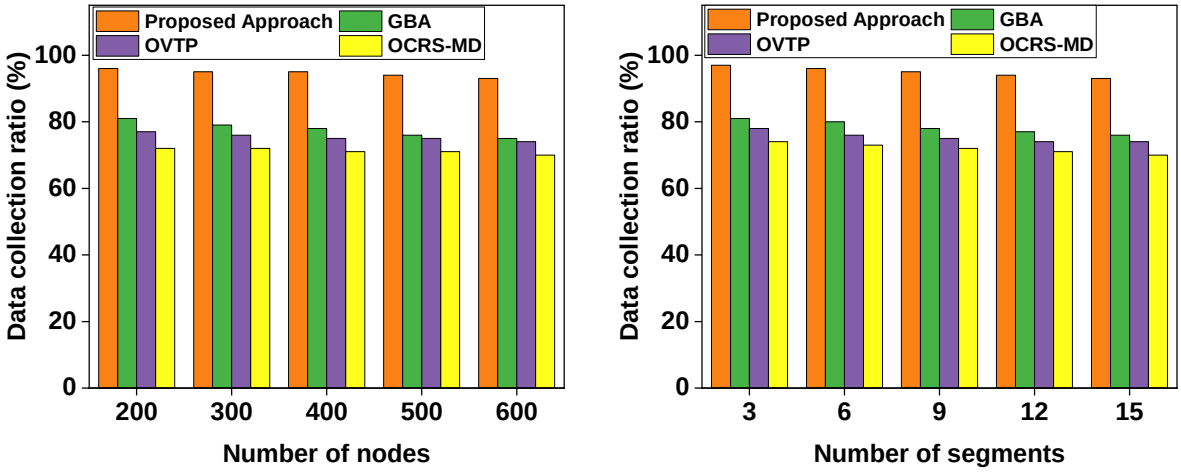


Figure 6.5: Data collection latency.

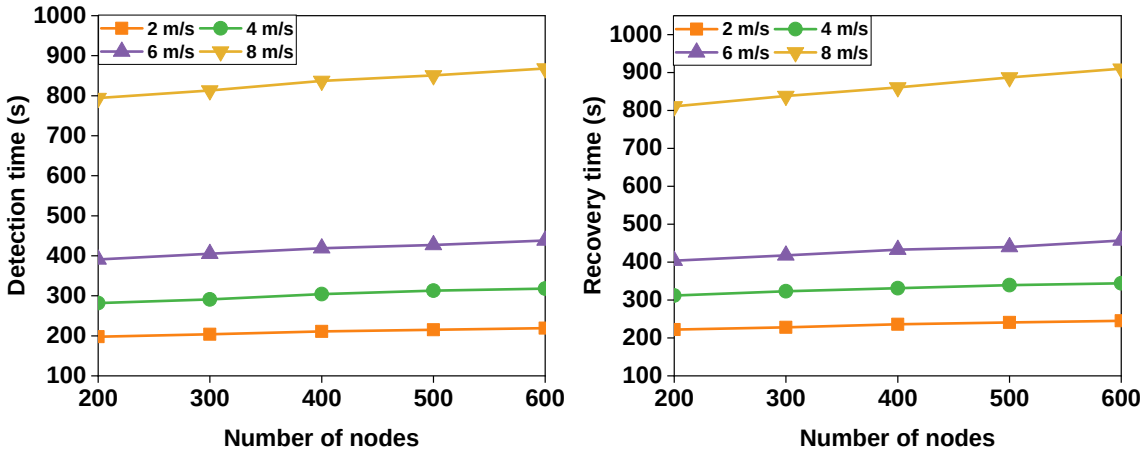
than OCRS-MD, 37% than OVTP, and 26% than GBA approaches. This is because the proposed approach selects the optimal RPs that are close to the cluster centre. It minimizes the average distance of DFNs with RP and helps to reduce the energy consumed in data transmission. The proposed approach also maximizes the node degree of RP that equally distributes the loads on DFNs and balances energy consumption throughout the cluster.



(a) (b)

Figure 6.6: Data collection ratio.

Fig. 6.5a shows the latency in different node densities. Fig. 6.5b shows the latency in different numbers of segments. The simulation result implies that the proposed approach reduces the latency by 43% than OCRS-MD, 38% than OVTP, and 28% than the GBA algorithm. This is because the proposed approach uses the RLBSO algorithm to select optimal RPs and design an optimal data-gathering path between the RPs. It minimizes the overall tour length. A shorter tour length reduces the data collection latency. Fig. 6.6a shows the ratio of data collection in different node densities. Fig. 6.6b shows the data collection ratio in different numbers of segments. The data collection ratio is the ratio of the amount of generated data by sensor nodes to the amount of collected data by MDC. The proposed approach shows the highest data collection ratio. It increases the data collection ratio by 32% as compared to OCRS-MD, 26% as compared to OVTP, and 22% as compared to GBA. This is because the proposed approach selects optimal RP positions and DFNs that enhance the data collection ratio. Furthermore, optimal path design using the RLBSO algorithm also improves the data collection ratio. The proposed approach is able to detect network cuts and perform recovery that minimizes the loss of data in the network. It helps to increase the data collection ratio.



(a) Detection time. (b) Recovery time.

Figure 6.7: Network cut detection and recovery.

Fig. 6.7a shows the average network cut detection time in different number of nodes. The detection time refers to the time duration between the occurrence of the network cut and the detection of the network cut. Fig. 6.7b shows the average recovery time in different node densities. Recovery time refers to the time duration between the initializing recovery process and the re-selection of RPs. The detection time and recovery time are high for the low speed of MDC and reduce with the increasing speed of MDC. This is because MDC takes a longer time to visit a cluster and perform network cut detection and recovery. Furthermore, Detection and recovery time also increase if a node becomes faulty after MDC has visited that cluster. This is because the network cut can only be detected after MDC completes its tour and visits that cluster in the next round.

6.5.1 Testbed

This section analyzes the performance of the proposed approach in real-life test best experiments. The experiment is performed in an outdoor environment, where eighteen sensor nodes are deployed in an area of 15x15 $[m^2]$ dimension. Fig. 6.8 shows the deployed nodes in the testbed environment. The sensor nodes used in the test bed

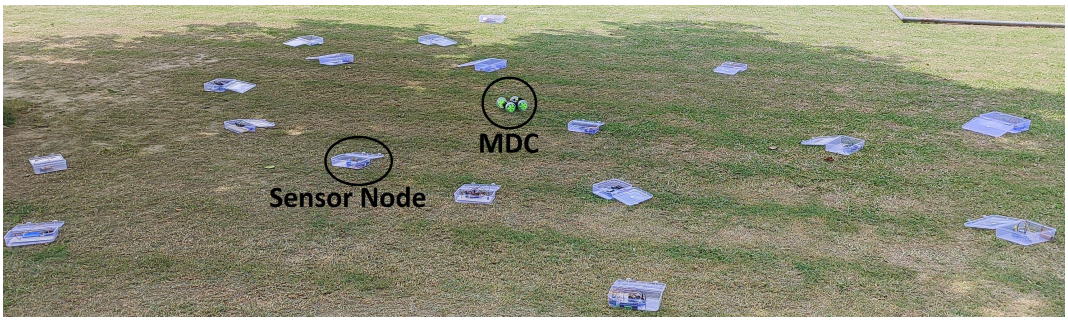


Figure 6.8: Sensor node deployed in the testbed.

Table 6.3: Average residual voltage (V)

Time (min)	*Propose Approach*	*GBA*	*OVTP*	*OCRS-MD*
30	10.84	10.07	9.62	9.56
60	9.72	8.49	7.98	7.86
90	8.58	6.87	6.33	6.14
120	7.43	5.19	4.58	4.33
150	5.94	3.56	2.79	2.31

consist of humidity, gas, and temperature sensors connected to an Arduino UNO board. In the sensor nodes, an XBee module is used for data transmission. Fig. 6.9 shows the node used in the testbed. A laptop is used as a BS, and a robotic car is used as an MDC.

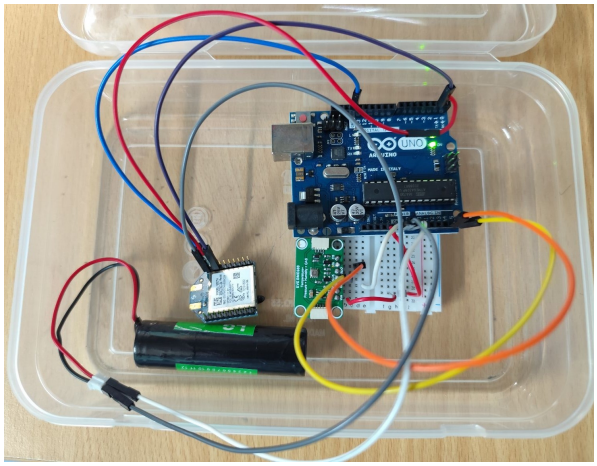
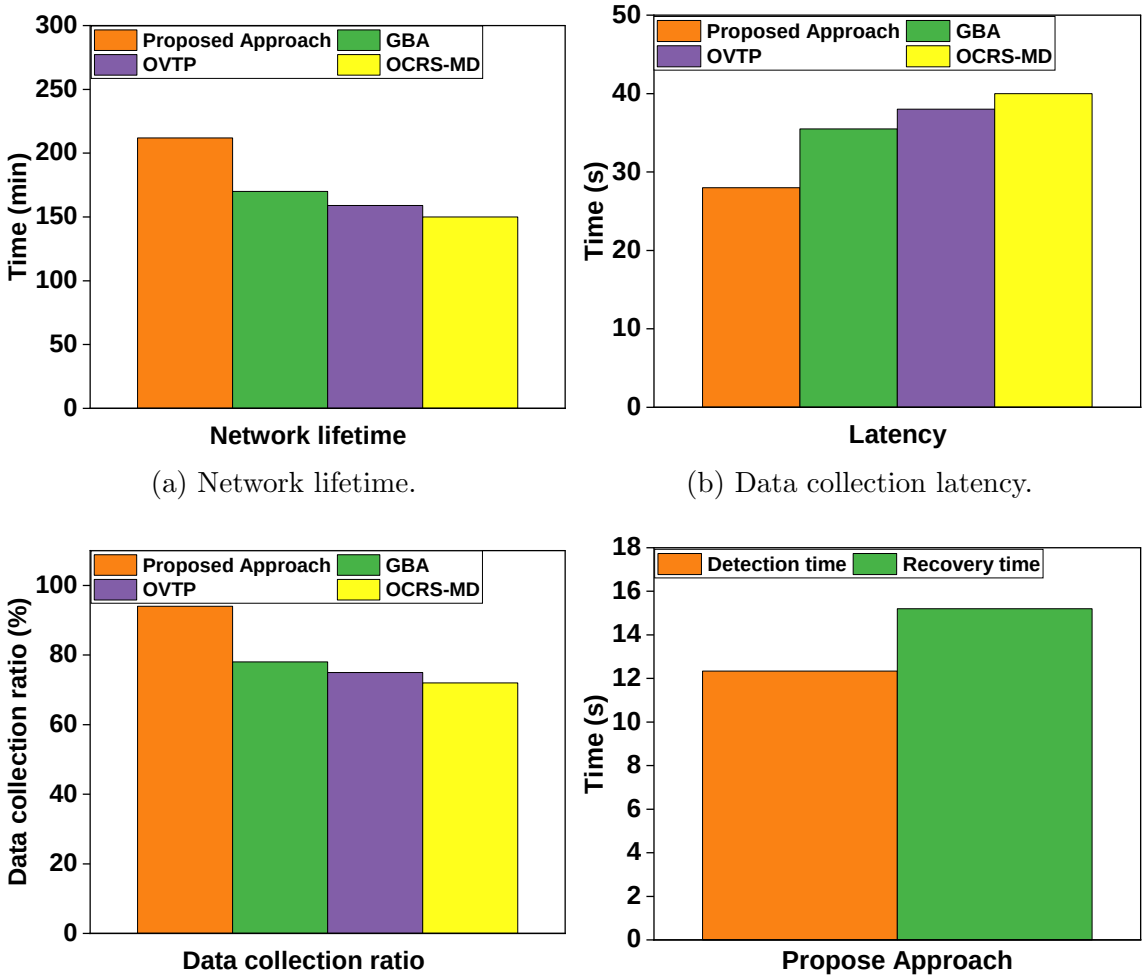


Figure 6.9: Sensor node used in the testbed.

Table 6.3 shows the average residual voltage of sensor nodes in different time intervals. The result shows that the proposed approach lessens energy expenditure up to 41% than OCRS-MD, 35% than OVTP, and 24% than GBA approaches. The result shows



(a) Network lifetime.

(b) Data collection latency.

(c) Data collection ratio.

(d) Detection and recovery time.

Figure 6.10: Testbed results.

that the proposed approach conserves more energy as compared to OCRS-MD, OVTP, and GBA approaches, similar to the simulation result. This is because of RLBSO algorithm-based optimal RP selection for data collection. The proposed approach minimizes the distance between RP and the cluster centre, which also reduces the sensor nodes' energy expenditure. Furthermore, the proposed approach selects DFN nodes that forward the data of sensor nodes to MDC. It saves energy at sensor nodes.

Fig. 6.10a shows the network lifetime. The proposed approach extends the network lifetime by 40% than OCRS-MD, 34% than OVTP, and 26% than GBA approaches. In the real environment, the proposed scheme depicts a higher network lifetime than

the other three approaches, similar to the simulation results. This is because the proposed approach maximizes the RP node degree, which divides the load equally among DFNs. It decreases the energy expenditure of DFNs and prolongs the network lifetime. Fig. 6.10b shows the data collection latency. The proposed approach reduces the data collection latency by 40% compared to OCRS-MD, 36% compared to OVTP, and 26% compared to GBA, similar to the simulation results. This is because the proposed approach reduces the length of the MDC path by an optimal number of RPs selection and optimal data gathering path design. A shorter MDC path results in reduced data collection delay. Fig. 6.10c shows the data collection ratio. The proposed approach shows an increase in data collection ratio by 30% compared to OCRS-MD, 24% compared to OVTP, and 20% compared to GBA, similar to simulation results. This is because the proposed approach selects optimal RPs and designs an optimal data collection path between the RPs, that increases the data collection ratio. Furthermore, the proposed approach effectively detects network cuts and performs recovery. It results in a high data collection ratio. Fig. 6.10d displays the network cut detection time and recovery time for the deployed testbed experiment. The result indicates that the proposed approach is able to perform recovery in considerably less time, similar to the simulation results.

6.6 Summary

In IoT-enabled WSNs, network cut is a major concern that severely degrades the performance of the WSNs. Therefore, this chapter proposes MDC-based novel network cut detection and network recovery algorithms. The proposed algorithm detects network cuts and the formation of isolated segments. It also performs recovery to enable the data collection from all of the isolated segments in WSNs. Furthermore, the proposed scheme applies the RLBSO algorithm to select the optimal number of RPs in the network and design the data-gathering path for MDC. The proposed scheme minimizes

data collection delay as well as energy consumption of sensor nodes. Simulation and testbed results indicate that the performance of the proposed scheme is superior as compared to the state-of-the-art approaches.

In the smart city, smart buildings play a vital role. Efficient emergency evacuation is one of the major challenging issues for any smart building. During a fire emergency, considering the dynamic spread of fire is very important while designing the emergency evacuation. Therefore, this thesis proposes a dynamic indoor emergency evacuation using IoT-enabled WSNs for smart buildings. It designs the shortest safe evacuation route for evacuees that significantly minimizes casualties and economic loss. The next chapter presents an indoor emergency evacuation system using IoT-enabled WSNs for smart buildings.