Improving the dependability of safety critical wireless sensor network scheduling using artificial intelligence

Al-Nader, I.

Improving the Dependability of Safety Critical Wireless Sensor Network Scheduling Using Artificial Intelligence

Issam Al-Nader

Supervised by

Prof. Aboubaker Lasebae

Dr Rand Raheem

Faculty of Sciences and Technology
Middlesex University
United Kingdom

January 2024

A thesis submitted to the faculty of Science and Technology at Middlesex University in partial fulfillment of the requirements for the degree of Doctorate of Philosophy in the Department of Computer Science

Abstract

To ensure optimal functionality and adherence to specified requirements, a Wireless Sensor Network (WSN) must prioritise the validation of its three fundamental attributes: Connectivity, Coverage, and Network Lifetime. Existing literature highlights numerous research endeavours to resolve reliability issues in WSNs, often concentrating on singular properties such as coverage and/or connectivity. These properties are frequently treated interchangeably and seldom concurrently addressed due to the intricate challenges arising from the distributed nature and resource constraints typical of WSNs. Moreover, safety critical WSNs applications require to follow rigid and strict policies such as strict deadlines while deploying in emergency situations which adds an additional layer of complexity. This layer of complexity is caused by new set of additional requirements not only connectivity, coverage, and lifetime. That is technically challenging in terms of optimizing the basic parameters required for the implementation of WSNs. Notably, there is a scarcity of published work comprehensively analysing and testing all three primary requirements; connectivity, coverage, and network lifetime of WSNs simultaneously, owing to their inherent complexity. This thesis tackled the three mentioned properties of safety critical WSNs as a Multi-Objective Optimisation (MOO) problem. The research methodology encompasses six key principles. Firstly, the Randomised Coverage-based Scheduling (RCS) algorithm is replicated, validated, and verified using a MATLAB simulation environment, revealing insights into node utilisation imbalances. Secondly, the performance of the RCS algorithm is scrutinised using the Perceptron Multilayer Artificial Neural Network (ANN) scheduling algorithm, exposing limitations. Thirdly, the Hidden Markov decision-process Model (HMM) is employed to enhance the service availability and reliability of the RCS algorithm, demonstrating superior optimisation metrics over RCS by increasing network lifetime while improving coverage and connectivity. Fourthly, a novel Bio-Inspired Bat algorithm was developed to address the identified limitations of previous scheduling algorithms, utilising objective optimisation functions and Pareto optimisation. The Bat algorithm outperformed the HMM algorithm across all metrics. Fifthly, the Self-Organising Feature Map (SOFM) algorithm surpasses the Bat algorithm with its straightforward approach to dimension reduction and classification. Sixthly, critical analyses of implemented algorithms (HMM, Bat, and SOFM) reveal similar patterns in

coverage data. Consequently, a Long Short-Term Memory (LSTM)-based node scheduling algorithm was introduced to analyse and provide an energy-efficient scheduling solution, addressing the Multi-Objective Optimization (MOO) highlighted in this research.

Table of Contents

List of Tables

List of Figures

List of Acronyms

A

A Lightweight Deployment-Aware Scheduling
(LDAS) 31

An Adaptive Self-Configuring Sensor Networks Topologies
(ASCENT) 30

C

Cluster Head
(CH) 36

D

Differential Evolution
(DE) 37

Distributed denial of services attack
(DOS) 49

E

Election-based Aquila Optimizer
(EAO) 36

Energy and Traffic Aware Sleep-Awake
(ETASA) 34

Energy Efficient Sleep Scheduling based on Application Semantics
(ESSAT) 32

Energy-Aware Scheduling Protocol
(ESP) 37

Energy-Efficient Sleep Scheduling algorithm
(ESSM) 40

G

Geographic Adaptive Fidelity
(GAF) 29

I

Intra-Cluster Distance
(ICD) 37

L

Learning Automata
(LA) 33

Lightweight Integrated Protocol Suite
(LIPS) 32

Long Short-Term Memory
(LSTM) 25

Acknowledgements

I wish to extend my heartfelt thanks to the most merciful Allah for providing me with guidance and patience during every phase of my PhD journey. My profound gratitude goes to my supervisor, Professor Abubaker Lasebae, for his unwavering support, invaluable guidance, and remarkable patience. I would also like to express my appreciation to Dr Rand Raheem for her steadfast support throughout this project. My sincere thanks are due to my family and especially my parents; Dr Flah Al-Nader and Ghzalah Al-Hamed for their enduring patience and unwavering support, which enabled me to successfully complete this study.

Declaration

This thesis has not previously been submitted in substance in full or part for any degree, and it is not concurrently submitted in candidature for any degree other than the degree of Philosophy of Doctorate of Middlesex University. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references. Certain parts of this thesis have appeared in published papers or are potential publications:

1. Al-Nader, I.; Lasebae, A.; Raheem, R. Using Hidden Markov Chain for Improving the Dependability of Safety-Critical WSNs'. In International Conference on Advanced Information Networking and Applications; Springer: Berlin/Heidelberg, Germany, 2023; pp. 460–472.

2. Al-Nader, I, Lasebae, A. and Raheem, R. A new perceptron-based neural-network algorithm to enhance the scheduling performance of safety-critical WSNs of increased dependability. 8th World Conference on Information Systems for Business Management. Bangkok, Thailand 07 - 08 Sep 2023 Springer.

3. Al-Nader, I.; Lasebae, A.; Raheem, R.; Khoshkholghi, A. A Novel Scheduling Algorithm for Improved Performance of Multi-Objective Safety-Critical Wireless Sensor Networks Using Long Short-Term Memory. Electronics 2023, 12, 4766, doi: https://doi.org/10.3390/electronics12234766.

4. Al-Nader, I.; Lasebae, A.; Raheem, R. A Novel Scheduling Algorithm for Improved Performance of Multi-Objective Safety-Critical WSN Using Spatial Self-Organizing Feature Map. Electronics 2023.

5. Al-Nader I, Lasebae A, Raheem R, Ekembe Ngondi G. A Novel Bio-Inspired Bat Node Scheduling Algorithm for Dependable Safety-Critical Wireless Sensor Network Systems. Sensors. 2024; 24(6):1928. https://doi.org/10.3390/s24061928

6. Al-Nader, I.; Lasebae, A.; Raheem, R. A Review for Node Scheduling Algorithm Using Machine Learning Approach. In the pipeline to be submitted to MDPI Sensors.

7. I. Alnader, A. Lasebae, R. Raheem “Using Hidden Markov Chain for Improving the Dependability of Safety-Critical WSNs", Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023). Lecture Notes in Networks and Systems, Springer eBook. pp.460–472, Volume 2. Online ISBN: 978-3-031-29056-5. https://doi.org/10.1007/978-3-031-29056-5_40, 2023 (Book contribution)

Chapter 1 Introduction

WSNs are embedded systems consisting of multiple distributed Sensor Nodes and one or more Base Stations, placed within an area of interest, to monitor and detect given behaviours and changes as shown in Figure 1.1 [Dener et al., 2022; Hill, 2003]. WSNs are used in a variety of monitoring, tracking, and control applications [Sohraby et al., 2007]. They make it easier to deploy applications in situations where infrastructure, such as communications and power, is scarce, especially for safety-critical applications. Safety-critical systems are concerned with the safety of human beings and their assets, and aim to avoid human injuries, deaths, economic loss, or environmental damage [Meier, 2009]. Hence, ensuring that a WSN functions correctly in the runtime is of paramount importance. WSNs are resource-constrained types of networks. Thereby, WSNs must be enabled with the ability to make a dynamic trade-off between energy consumption, application requirements and the system performance with respect to dependability requirements.

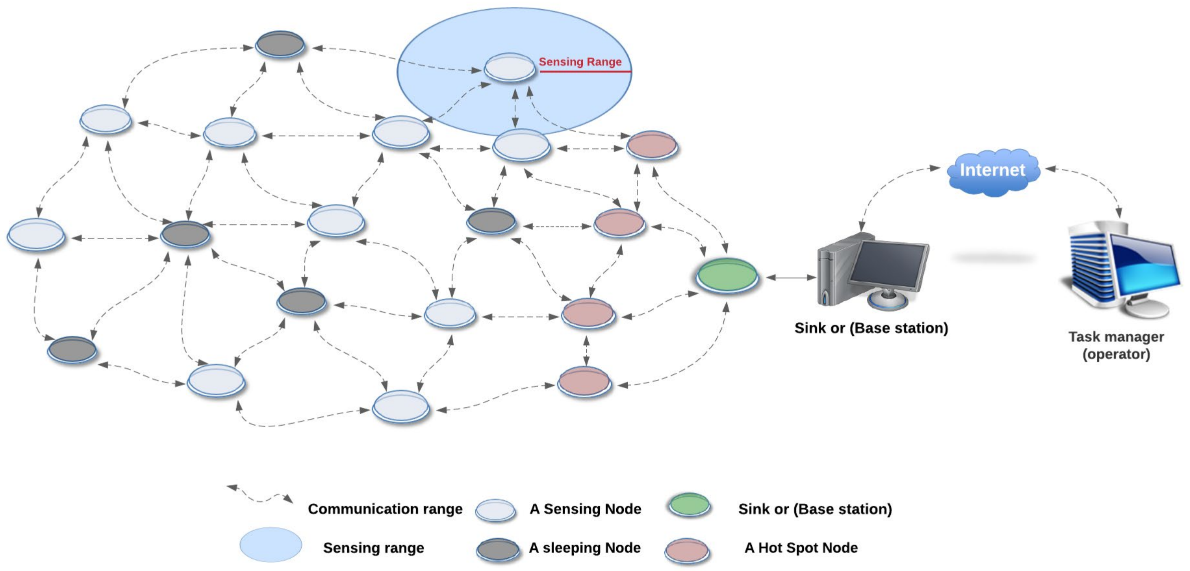


Figure 1.1 Demonstration of the WSN Structure

WSN systems must be made dependable, where their dependability is determined by three primary properties: Coverage, Connectivity and Network lifetime [Sohraby et al., 2007]. Coverage is concerned with the ability to detect events in the area to be monitored; Connectivity is the ability to communicate the detected events; and lifetime is the length of time a WSN needs to perform its functions [Tian and Georganas, 2002; Ye et al., 2003]. Figure 1.2 illustrates the increased usage of the WSN between 2015 and 2025 [Dener et al., 2022].

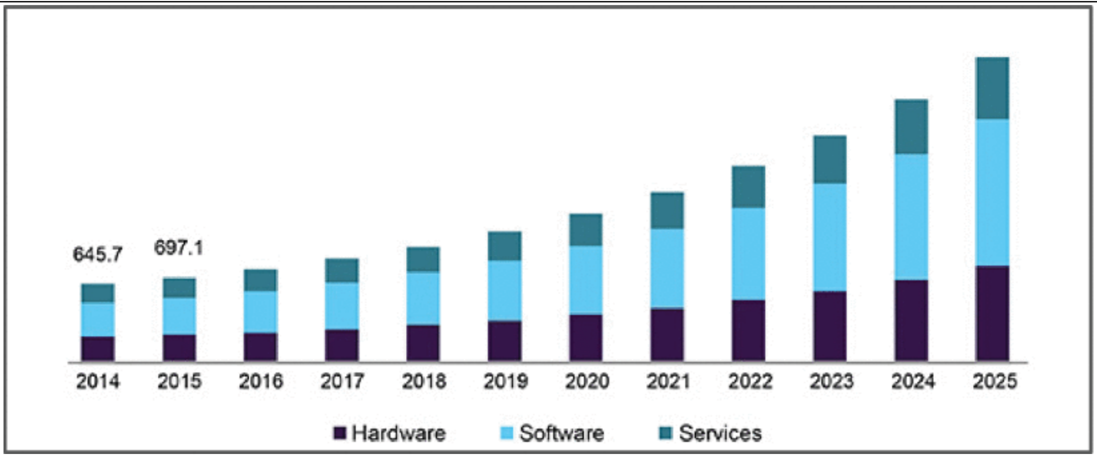


Figure 1.2 The Industrial WSN market size, 2015 -2025 (USD Million)

The dependability requirements such as reliability and availability for WSNs must be addressed to achieve the level of Quality of Service (QoS) required by the application. [Hartung et al., 2006]. Several factors can impact the dependability of the services in WSNs such as the harsh environments they are deployed in, the wireless link they communicate with, and their limited energy supply (via batteries) they are equipped with [Ammari and Das, 2006]. Thus, what follows is an overview of the factors that impact the dependability of the WSNs.

1.1 An Overview of WSNs

A WSN has two main elements: nodes and a base station. WSN nodes operate with limited resources, e.g., limited communication range, limited processing, and limited power supply. The role of a node is to sense the surrounding environment (Coverage) and communicate with other nodes to report the sensed data to the base station (Connectivity). The maximal sensing area of a node is called the sensing range, that is, the coverage area of the sensor node [Tian and Georganas, 2002; Ye et al., 2003]. The area covered by the whole network or network coverage is determined by the coverage areas of all of its nodes. The maximal communication radius of a node is called a communication link. A node is connected to the base station if there is a communication path or link from the node to the base station. Let A and B be two nodes. If node A is within the communication range of B, A is a neighbour of B; conversely, B is a neighbour of A.

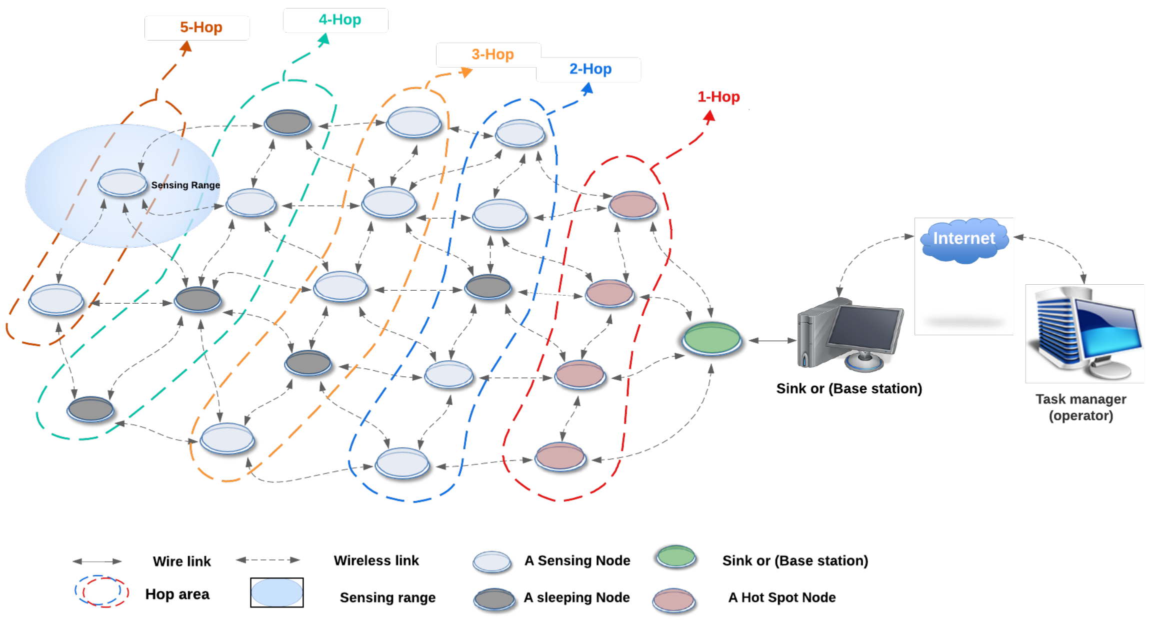


Figure 1.3 Demonstration of the WSN elements of dependability.

In this work, to study the elements of dependability in WSNs, a WSN will be divided into *Hop areas* whereby nodes that are 1 Hop count away from the Base station belong in 1-Hop area; nodes that are 2 Hop counts are in 2- Hop areas and so forth as illustrated by Figure 1.3

The following are important element to study the dependability of WSN:

- Definition 1 (Nodes). A node is a device that senses its environments in order to detect given events.
- Definition 2 (Base station). A base station is a special node that collects data sensed by other nodes and typically has an infinite lifetime.
- Definition 3 (Downstream). A communication is downstream if it flows from the base station to the nodes. If a node is receiving information from (the direction of) the base station, this node is called a downstream node. Let A and B be two nodes. If A sends a message downstream to B, then B is called a downstream node of A.
- Definition 4 (Upstream). A communication is upstream if it flows from any node to the base station. If a node is sending information to the base station, then this node is an upstream node. Let A and B be two nodes. If B sends a message upstream to A, then A is called an upstream node of B.
- Definition 5 (Neighbours). The neighbours of a node are all the nodes within its communication range. Let neighbours(A) denote the neighbours of node A.

- Definition 6 (Minimum hop count). The minimum hop count of a node is the shortest path to the base station, where a path is the number of (intermediate) nodes from a given node to the base station.
- Definition 7 (Rounds). All the processes that occur within a single iteration cycle in regard to system process scheduling algorithms It is the time where all the sensor nodes send data to the base station.

Dependability of WSN: According to [Xu et al., 2001], the dependability of a system is the ability of the system to avoid service failures that are more frequent or more severe than is acceptable. The principal threats to the dependability of WSNs stem from the failures relating to energy efficiency [Busse, 2008] [He et al., 2004] [Hartung et al., 2006]. One must also consider failures linked to WSN components. **Node failure** can be due to a malfunction or a breakdown in one of its components, e.g., memory, processor, sensor, and battery. **Communication (link) failure** can be due to many reasons, e.g., radio interference, collision, or due to a node dying as a result of fully depleting its energy. **Coverage failure** occurs when a node fails to collect data in its target area due to battery depletion. Coverage and communication go together. If a node covers an area but cannot communicate back to the Base Station, this is as if the area is not covered. Reciprocally, if a node can communicate back to its Base Station but there is no coverage, then events cannot be detected, as if there is no communication.

In this work, a WSN is dependable for a given application if its maximum lifetime is equal or greater than its expected service time. In addition, a WSN must be rigorously tested to state its dependability.

A WSN has a number of performance challenges such as connectivity compromise, coverage holes, network lifetime reduction, throughput delay, and reduced data volume in a network. If such challenges are mitigated, then a dependable WSN can be achieved. Hereafter is a deeper dive into such challenges as they become the focus of the multi-objective optimization work presented in this thesis.

Connectivity: A WSN is required to report back data captured from sensors to the base station. Hence, connectivity at a global level (from all nodes to the base station) is considered important. For example, consider a WSN with all nodes connected - if a link failure happens, where two neighbouring nodes are disconnected, may lead to partition the network into different sub-networks. Where one sub-network (X) is connected to the base station, while the

other (Y) is not. Then events in Y will not be reported, thus affecting the dependability (service availability and reliability) of the whole WSN.
The notion of ‘k-connectivity’ has been used to address the connectivity problem in WSNs and will be adopted throughout the course of this thesis. For example, a network has ‘1-connectivity’ means that there exists one link between two nodes. A WSN is said to be connected when there is at least one connection between all its nodes [Ammari and Das, 2006].

Coverage: Coverage is concerned with the physical distribution of nodes in the area to be monitored. Coverage is an essential attribute for WSNs because it affects the way events are detected. What is enough coverage? To answer this question, an explanation of how coverage is measured in WSNs is needed. As defined by [Tian and Georganas, 2002; Ye et al., 2003], "the degree of coverage at a particular point in the sensing field can be related to the number of sensors whose sensing range covers that point".

In this thesis we will use ‘C-coverage’, where C is the number of sensors e.g., ‘1-coverage’ means that a point in the area to be monitored is covered by the sensing range of only one active sensor node. Then, if an area is covered by one sensor node, then the degree coverage of that area is 1.
The degree of sensing coverage varies from an application to another. While in some applications the degree of sensing coverage is required to be high, e.g., military applications, in other applications it might be low, e.g., animal and habit monitoring. Other applications may require a dynamic configuration from low to a high degree of sensing coverage, e.g., intrusion detection system [Wang et al., 2003]. Thus, a WSN that has a higher degree of sensing coverage will accurately sense and capture events, and therefore will be more robust to potential sensing failure [Cerpa and Estrin, 2004].
Moreover, in this thesis the sensing coverage is defined as a function of at least three main parameters: node presence or the location / deployment of a node in a point of interest in the monitored area; node lifetime or the battery life expectancy of the node; node malfunction or the healthy performance of nodes without defects / errors / faults. In this thesis, a WSN is covered if at least the point of interest which is part of the WSN monitored area is covered by one, healthy (not malfunctioning) and alive (known lifetime) node.
Network lifetime: It is the time during which a WSN is able to sense and communicate data to the base station. The network lifetime of the WSN is affected by the number of communications (connectivity) and the sensing (coverage) activities where events are detected

and reported due to energy consumption to perform such activities. There is a trade-off between cost and performance in any WSN. The cost is energy consumption, and the performance is manifested by coverage as well as connectivity. A WSN may have all sensors switched ON at all times, maximizing the coverage and the connectivity factors, but that would consume more energy. Energy consumption is a key metric that can provide an estimate of the WSN lifetime.

There are multiple definitions of the lifetime in the literature: "the time till the first sensor node runs out of its battery energy" [Xu et al., 2001]; "the total number of nodes alive over time" [Cardei et al., 2005]; "the time until a given percentage of nodes in the WSN goes below a given threshold" [Chiang, 2007].

In this thesis, network lifetime is the longest (for availability), useful (for always being reliable), and meaningful (for corruption-free data transferred) network lifetime.

There are several approaches to ensure the dependability of a resource-constrained WSN [Thomas et al., 2020]. One approach is having an endless power source in each node, but since sensor nodes operate on a limited battery, inherently, this solution is deemed impossible. Another approach is scheduling nodes to sleep and wake up to preserve the energy [Tian and Georganas, 2002]. However, a node that is scheduled to sleep loses its connectivity and coverage and no longer participates in the network activity. This can introduce a coverage hole whereby critical events cannot be detected, or a connectivity hole whereby detected events are not being routed to the Base Station. In resource constraints WSN this is known as the MOO [Liu et al., 2006], [Wang et al., 2010]; or the M-connected coverage MOO problem [Mini et al., 2012].

Several works have attempted to solve the MOO problem, however, they consider one or two objectives only, e.g., [Ambrose, 2008; Cerpa and Estrin, 2004; Chipara, 2010; Ha, 2006; He et al., 2004; Tate and Bate, 2011; Viera et al., 2003; Wu et al., 2005; Xu et al., 2001; Yang, 2011; Ye et al., 2003] do not consider network coverage in their works; other works such as [Tian and Georganas, 2002] and [Cardei et al., 2005] consider network coverage, but without ensuring a good level of network connectivity. Only a few works address the MOO problem considering the three objectives: Coverage, connectivity and network lifetime at the same time, e.g., [Liu et al., 2006], [Wang et al., 2010].

[Liu et al., 2006] propose the RCS algorithm which splits the network into different sub-networks where Nodes join each of these sub-networks randomly. These sub-networks alternate between ON and OFF states (a subnetwork is ON/OFF meaning that all of its nodes are ON/OFF). An issue with RCS is that it forces nodes to be turned ON although they are scheduled to be in an OFF state, thus causing nodes to consume energy faster and die prematurely. Another issue is that Nodes which are closer to the Base Station are more likely to be used than others. This will lead to the partition of the network where events can be detected but not reported back to the Bases Station.

In this thesis, the main concern was to address the limitations of the RCS algorithm and provide an optimal solution for the MOO problem. Therefore, First, the state of the WSN needs to be extensively analysed. Second, the search space must be fully explored to obtain the optimal solution.

1.2 Problem Definition

Examining the work of [Liu et al., 2006], there is a necessity to thoroughly investigate the (solution space for identifying the optimal schedule. In particular, investigate how the parameters like connectivity, coverage and network lifetime would manage to provide the best duty cycle preserving the energy and topology of the WSN. The effectiveness of the proposed RCS algorithm is not evident. Considering that sensor nodes near the base station may experience earlier depletion due to increased workloads, it is conceivable that nodes in a 2-Hop area could potentially offset the workload of a node in a 1-Hop area. While increased energy consumption is anticipated for the backup nodes due to the extended communication distance, this remains a more favourable option compared to network partitioning. Consequently, it is imperative to delve into the solution space. A method for comparing the RCS original schedule with alternative scheduling algorithms is essential for determining the most optimal schedule. Regrettably, RCS lacks the capability for such a comparison on its own. Only through this comparative analysis one can ascertain the optimal solution.

The solution space for RCS scheduling algorithm is very large. One way of finding best schedules is by feeding each one of them into a simulator and see which one works best. Using a WSN simulation model where we can exactly show the best scheduling with the help of scheduling and optimization processes that are built in the simulator.

In this work we propose exploring alternative ways to represent the solution space that make it easier to explore a larger portion of the solution space and select the optimal solution. In the literature multiple approaches are possible including Hidden Markov Model, Bio-inspired computation, and machine learning. Due to the distinct representations of the problem offered by each scheduling algorithms, it is impossible to determine in advance which one will yield the optimal result. This is because each problem has its own new presentation for the solution. Consequently, the approach involves (1) employing each algorithm separately to identify the most effective schedule and subsequently (2) comparing the outcomes of each approach through simulation. Furthermore, one will assess the user-friendliness of each approach and examine the ease of verification for each.

1.3 Research Questions

The hypothesis of the thesis is:
"There is a mechanism for studying, analysing the complexity of safety critical WSNs to improve connectivity, coverage, and network lifetime. Such analysis will improve the service availability and reliability of dependable safety critical WSNs."

From this hypothesis, the following research questions have been derived which were addressed in this thesis's technical chapters as illustrated below:

1. Can Neural Networks ANN analyse the complex behaviour of WSNs' scheduling algorithms to uncover limitations to improve performance? Chapter 3 addresses this question with the novel artificial neural network ANN analyser.
2. Is there a method for improving the previous work in the previous research question using machine learning AI algorithms? Chapter 4 addresses this question with the novel HMM utilisation of the RCS algorithm.
3. Can Bio-inspired algorithms improve on the work achieved in previous chapters and research questions with respect to the multi-objective optimisation problems? Chapter 5 addresses this question with the novel Bat algorithm.
4. Can the aforementioned scheduling improvements be maintained with simple design principles to mitigate network overheads? Chapter 6 addresses this question with the novel application of the SOFM algorithm in safety critical WSNs.

5. Is there a scheduling algorithm that can address the limitations of the previously analysed scheduling algorithms? Chapter 7 addresses this question with the proposed novel application of the LSTM algorithm.

1.4 Aims and Objectives

The main **aim** is to address the MOO problem in WSNs [Liu et al., 2006], [Wang et al., 2010], [Thomas et al., 2020] scheduling algorithm. **Wherein, by optimising the network requirements will improve the dependability (service availability and reliability) of the WSNs**. Hence, the main task is to improve the key functional network requirements of a WSN namely, lifetime, coverage, and connectivity via the use of Machine learning and Bio-inspired methods in the Node scheduling approach.

In order to meet the aim as mentioned above the following objectives will be implemented as follows:

- The ANN algorithm analyses the complex behaviour of the WSNs during their run time. Thereby, capture limitations that when addressed shall improve the performance of the WSNs.
- The HMM Model is utilised to optimises the nodes Sending and Receiving states whereby, an energy efficient HMM based scheduling algorithm is proposed to improve upon the three network requirements (lifetime, coverage, and connectivity) to achieve optimal QoS.
- The Bio-inspired Bat based scheduling algorithm is built to optimize the node scheduling of WSNs along the three requirements of coverage, connectivity, and lifetime. This notably permits the exploration of a larger search space to find the best scheduling solution. Moreover, the Bat algorithm is coupled with the filtering power of the pareto algorithm where it is used to provide more accurate values of energy of sensor nodes to provide better QoS of the network.
- The (SOFM and LSTM) Machine learning based scheduling algorithms were used to optimise and achieve optimal QoS. Several proposed scheduling solutions in the literature kept out critical design considerations such as network coverage and connectivity.

Therefore, this thesis is to look into these challenges and to find an optimal solution to reconcile these three network requirements which are essential to resource constrained WSNs.

1.5 Thesis Contributions

In order to find the solution to the MOO problem, the following **contributions** of the optimal node schedule are listed below:

Contribution: **A Perceptron-based Artificial Neural Network (ANN) Analyser**. The ANN analyse, learn, train, and test the performance of scheduling algorithms and returns the number of times for each node working schedule (Overworked/Overused) to better the overall dependability in WSN. The Analysis of the RCS by [Liu et al., 2006] using the ANN analyser confirms the initial Hypothesis, that is nodes which are closer to the Base Station are more likely to be used than others. The ANN analysis shows the limitation of the RCS algorithm in which the network will be partitioned. The contribution of the ANN analyser are two folds:

- A model for a WSN, comparable to the one defined in by [Liu et al., 2006].
- A Dataset Model for a WSN scheduling algorithm.

Contribution: **A New HMM-based scheduling algorithm**. The HMM algorithm show that it improves the lifetime of the network by detecting nodes that will die early and rebalancing their workload. The HMM technique can also be used for diagnosis and provide maintenance warnings to WSN system administrators. Ultimately, the HMM technique can be used to improve algorithms other than the RCS.

The hypothesis was validated by applying the Hidden Markov Model, the HMM solution achieve two main tasks:

- Analyse the duty cycle of every node (Sleep-Awake /Send-Receive) state locally.
- Minimise the sending state and maximise the receiving state to balance the workload of the node duty cycle locally.

Since, the RCS scheduling algorithm does not explore the state space of the problem. HMM scheduling algorithm explore the information made available by the nodes, but it is stuck in a local minima and local maxima.

Contribution: A New Bio-inspired Bat based Scheduling Algorithm. A number of problems related to explore the search space globally have been solved.

- The Bat algorithm defines a fitness and objective function over a search space which returns all possible sleep and wake up schedules for each node in the WSN. This yields a (scheduling) solution space that is then organized by Pareto sorting whose output coordinates are the distance of each node to the Base station and the residual energy of the node shows significant improvements on all three (3) dimensions, and Pareto sorting guarantees us that the proposed Bat scheduling is the best, for all possible fitness and objective functions and for a given search space. The Bat algorithm has limited scalability so when parameters is less it performs reasonably well, but it falters when number of parameters is large. The Bat Algorithm relies on randomness in its search process, which means that its performance can vary from one run to another. The Bat algorithm converges slowly due to its computational complexity. While the computation is heavy, it takes shorter path to achieve the result avoiding the global search space.

Contribution: A New Machine learning-based scheduling algorithms. A number of problems related to explore different representations of the solution space have been solved.

A New Self-Organising Feature Map (SOFM) spatial-based scheduling algorithm is proposed to a achieve an optimal scheduling solution. The proposed SOFM algorithm is based on the Self-Organising Map (SOM) model that uses variables such as, 'The position of the cluster head to the Sink, the distances between the parent to child nodes, the Energy levels, Tx (transmission distance) and Rx (receiving distance)' which do not change over time. The SOFM algorithm visualise the desired topology based on the input variables above where the optimal lifetime coverage and connectivity is attained. In another words, the SOFM algorithm summarizing the data in a lower-dimensional space to facilitate easier analysis and understanding of the network behaviour hence obtain the new topology. The limitation of SOFM approach is that once the predefined features are considered, the scheduling algorithm will be limited to these features. This limitation can be addressed using the LSTM model, where selection and/or adding/deleting of features are possible.

- A New Long Short-Term Memory (LSTM) time-based scheduling algorithm. The LSTM algorithm achieves the energy-efficient scheduling solution, by replicating a pattern of high value so that the nodes in the networks adapt the same performance in all the rounds. The LSTM algorithm confirms our hypothesis by finding an efficient sequence of coverage with improved coverage without compromising network connectivity and lifetime. The LSTM time-based node scheduling algorithm limitations are: (1) there is an issue with the long chain of sequences in managing the best possible cycle. (2) the LSTM is computationally intensive due to long chains, (3) data requires are comparatively large to find the better to best (more data the better the results) and, (4) LSTM has limited memory processing.

1.6 Thesis Scope

Many solutions attempt to address the MOO problem; however, they are not all optimal in their search space in the sense that given a node scheduling algorithm such as [Liu et al., 2006], it often remains possible to find a better schedule showing that the proposed solution is not the best, even when it is better compared to a distinct algorithm. Therefore, this thesis looks at different models to find the best solution within their search space, i.e., within these models, the proposed solution is considered to be the best one. The literature provides many optimisation approaches for WSNs optimisation such as Bio-Inspired Computation, Evolutionary, Genetic, Machine Learning, and Node Scheduling Algorithms. This thesis focuses on Node Scheduling, where the use of HMM, Bat Algorithms, Self- SOFM, and LSTM as models to build node scheduling algorithms and provide optimal schedules within their respective search space have been explored. Also, the ANN analyser is used as a tool to for a given algorithm to analyse and tell if the algorithm's proposed solution is optimal. This analysis technique is applied against the [Liu et al., 2006] algorithm and found that 1-Hop area nodes would all die first. It is difficult, however, to generalise the ANN-based analysis because different algorithms have a different state space, making it necessary to build a new ANN-analyser each time. It also did not seem feasible within this timeline to try and build a general-purpose analyser say for all Bat Algorithms, or for all HMMs. Whilst it is more difficult to provide guarantees for very complex algorithms, building a general-purpose analysis tool capable of certifying that a given algorithm, independently of its underlying model and state space is optimal, is out of the scope of this thesis. Thus, to find the best solution within different search spaces, the simulation results of the proposed algorithms have been compared. Also, the complexity of the model and possible shortfalls as elements to consider before choosing one model or another have been evaluated. Since some models add only complexity with little added performance value, e.g., SOFM is more complex than LSTM but offers similar performance results which could be future work consideration. Obviously, a more systematic study needs to be realised to reach a more definitive conclusion. Other methods could be envisaged, notably testing, whereby a single specification could be used to test different algorithms, however, differences in search space would mean building different test cases or being able to translate some initial test case into another. Alternatively, the models themselves could be compared and see if they could be mapped /translated say a Bat algorithm into a SOFM model.

1.7 Thesis Outline

This section highlights the structure of this thesis. The chapter titles are listed below with a brief introduction for each one.

Chapter 2: Literature Review
In this chapter, the latest published literature of state-of-the-art node scheduling of WSNs was reviewed that including the ANN, HMM, and the Bio-inspired BAT works. Such areas of research will be defined in Chapter 2 to introduce the topics before reviewing the related works. The optimization in this work depends on a new definition of WSN dependability which is introduced in Chapter 3.
The Chapter has also included relevant and similar work published in SOFM of which the author of this thesis has developed a new scheduling WSN algorithm which has addressed the multi-objective WSN scheduling optimisation problem in a novel way.

Chapter 3: A Perceptron-Based Neural-Network Analyser to Enhance the Scheduling Performance
The ANN algorithm is used to inspect and analyse the RCS algorithm’s behaviour and performance with respect to the discrepancy in the nodes being scheduled ON or OFF. This is critical in order to increase coverage, connection, longevity, and hence WSN dependability.

The additional benefit of the presented ANN-based analyser is its extended scalability of implementing the proposed analyser in all WSN scheduling algorithms, not limited to RCS per se.

Chapter 4: Using Hidden Markov Chain for Improving the Dependability of Safety-Critical WSNs

The HMM HMM-based scheduling algorithm is proposed to optimise duty cycles assigned to nodes with respect to their distance from the base station as well as the overall number of nodes in the network. Using HMM, the base station is able to predict the On and Off states for each node, hence an optimal number of duty cycles can be deployed to each node to ensure coverage, connectivity as well as increased network life longevity.

Chapter 5: A Novel Implementation of a Bio-inspired Bat WSN Scheduling Algorithm

The Bat algorithm is proposed as an improved solution to the HMM in which the former does not require randomized seed sequencing in its search space optimization. The BAT algorithm uses Bio-inspired techniques for its search space without the need for computational overheads where the new results is obtained at a global level.

Chapter 6: A New WSN Scheduling Improvement via the Self-Organizing Feature Map (SOFM)

In this chapter, a SOFM is used to enhance the scheduling results obtained from previous chapters which allows a complex ad-hoc network to be represented in a hierarchical manner, allowing SOM optimisations to work with lower computational complexity to meet the WSN scheduling objectives. The advantages associated with SOM implementation are reduced complexity and a simple depiction of the scheduling results. The novelty of the work presented in this chapter stems from utilising SOFM in WSN scheduling to improve the analysis of the results.

Chapter 7: A New Scheduling Algorithm for Improved Performance of Multi-Objective Safety-Critical WSN Using LSTM

Given the critical analyses of the previously implemented algorithms namely HMM, Bat and SOFM notable observations have yielded the detection of similar patterns in our coverage data.

Therefore, an LSTM-based algorithm was introduced to analyse and provide a suitable, energy-efficient scheduling solution that addresses the MOO problem.

Chapter 8: Conclusions and Future Work

This chapter presents an evaluation, conclusion, and a summary of the proposed work. Furthermore, a future work is presented in this thesis.

Chapter 2 Literature Review

This chapter will include the literature review for the upcoming technical contributions discussed in the individual chapters. A great deal of attention will be given to node scheduling approaches [Thomas et al., 2020] as it is the main area of focus. The reviewed works, to the author’s best knowledge, confirm that the present problems within WSNs go beyond energy efficiency, other areas of focus should also include coverage and connectivity of WSNs.

2.1 Node Scheduling Approaches

Most node scheduling algorithms in the literature do not address the three requirements of the WSN altogether, i.e., coverage, connectivity, and network lifetime. In particular, most algorithms address each requirement in isolation or make unrealistic assumptions like ideal conditions of WSN, no coverage considered, etc. [Xu et al., 2001] proposed a scheduling algorithm called Geographic Adaptive Fidelity (GAF). GAF saves energy by assuming sensor nodes maintain a good level of connectivity. This presumption poses a limitation since the communication links between nodes can be susceptible to various factors impacting the scheduling algorithm's performance. Consequently, their suggested solution becomes impractical for Wireless Sensor Networks (WSNs) with safety-critical applications such as forest fire detection systems. Additionally, GAF overlooks network coverage, primarily due to nodes being powered down (scheduled to sleep) by deactivating their communication units, resulting in a blind spot.

The scheduling algorithm proposed by [Tian and Georganas, 2002] examines the relationship between energy conservation and the time it takes for a node to determine its eligibility for being powered down (asleep). The faster a node makes this decision, the greater the energy savings. However, this approach is plagued by control message overheads, as it necessitates the use of GPS for obtaining geo-location information of sensor nodes. While convenient, this method is costly. As networks age over time, neither network connectivity nor coverage is maintained. In the event of a node failure (due to depleted energy), the dormant nodes erroneously assume the failed nodes are still active. Consequently, the network becomes partitioned, isolating some nodes. Tian et al.'s approach concentrates solely on network connectivity and energy efficiency, neglecting the aspect of coverage.

[Ye et al., 2003] developed a robust energy-conserving algorithm for long-lived sensor networks called Probing Environment and Adaptive Sleeping (PEAS). The algorithm tends to maintain network coverage by activating nodes that do not receive a reply to the probe message. This approach also suffers control message overheads due to broadcasting the probe messages periodically to maintain a set of active nodes. This approach is ineffective, when a node is designated as active, it sustains activity until its energy is completely depleted. Consequently, this results in a coverage gap within the network. This method does not guarantee the preservation of the initial sensing coverage, potentially causing significant events to go unnoticed and unreported to the base station.

[Viera et al., 2003] developed a scheduling algorithm based on the Voronoi diagram. The algorithm extends the network lifetime by exploiting node redundancy. Once a node is determined active and responsible for monitoring an area of interest it stays active until its energy is depleted. The paper fails to specify the timeframe for reactivating the powered-down nodes, and it does not discuss the effectiveness of coverage maintenance. Even if nodes within a limited area are designated to be turned off, the challenge of addressing network coverage is not adequately addressed.

An Adaptive Self-Configuring Sensor Networks Topologies (ASCENT) algorithm is presented by [Cerpa and Estrin, 2004]. ASCENT has several drawbacks. First, it demands each node to be aware of the network state information resulting in control message overheads. Second, the authors did not consider network coverage in the design of this solution. Finally, having two states (passive and test) to alternate between active and sleep states, requires additional waiting time and data collection in each of these states; consequently; this affects ASCENT's ability to adapt to network dynamics rapidly. In addition, once nodes become active, they stay active till they deplete energy, this results in portioning the WSNs.

The authors in [He et al., 2004] proposed a distributed scheduling algorithm. The proposed solution suffers from a control message overhead. Another drawback is in its reactive approach where sensor nodes may drift in time. As a result, sentry (master) nodes may not know for certain which nodes are asleep and which are awake. Hence, the non-sentry nodes will keep broadcasting multiple wake-up beacon messages to all non-sentries in the network.

While the authors in [Cardei et al., 2005] proposed a scheduling algorithm which organises the network's sensor nodes network into sets. Each set covers the target area at different time intervals. Finding the maximum number of sets can provide good network coverage and extend the network lifetime. This approach has limitations; First, it requires the geo-location of each sensor node which is very expensive. Many applications do not require the use of the GPS. In addition, the solution focuses on providing good sensing coverage while omitting network connectivity. Thus, important events may be detected but not reported back to the base station for processing, which is meaningless.

A Lightweight Deployment-Aware Scheduling (LDAS) presented by [Wu et al., 2005] assumes a sensor node maintains three states: on-duty, ready-to-off, and off-duty to improve the network lifetime as part of its scheduling algorithm. LDAS has managed to extend the network lifetime, but it didn't consider network coverage.

Another interesting sleep scheduling algorithm proposed by [Ha, 2006] is based on spanning tree topology which is used to manage energy consumption and extend the network lifetime of the WSN. The algorithm is susceptible to parent node failures. For example, the failure of a parent node would cut off the communication with its sub-tree nodes (children) affecting the end-to-end communication reliability. Such failure would partition the WSN even if it remains connected. The use of tree-based routing achieves energy efficiency but at the cost of delay because data will always be reported via a single route. This prevents promptness and timing performance of reporting data to the sink node at the appropriate desired time.

Another study proposed by [Ambrose, 2008] in which a distributed scheduling algorithm for composite events detection in WSNs to solve multi-objective problems concerning coverage and energy consumption was introduced. A composite event detection is an event that consists of two or more simple events. This work tends to save energy by (1) scheduling nodes to sleep, and (2) minimising the transmission range between each node in the network while maintaining network connectivity. The author assumed that node locations are known either by GPS or other computation localisation methods. This assumption is unrealistic and cost-efficient in large-scale WSNs. The deployment environment in the paper was a simple grid area, but large-scale WSNs usually consist of complex and dynamic obstacles. The distributed scheduling algorithm implies a complex computation process, especially when maintaining network connectivity, which causes an energy overhead. In a busy WSN, such a computation is not acceptable, hence the need for a simple computation algorithm is highly sought after in resource constrained WSNs.

Another equally important study by [Chipara, 2010] proposed Energy Efficient Sleep Scheduling algorithm based on the Application Semantics (ESSAT) model which introduces a sleep schedule that considers both energy consumption and the application deadline. ESSAT consists of two main components: a traffic shaper and a sleep schedule. ESSAT addressed a challenging optimisation problem, but it depends on the prediction of the time properties of the workload information that is obtained from earlier message exchanges amongst sensor nodes. ESSAT is in turn vulnerable and dependent on time. However, an advantage is that the sleep schedule requires no time synchronisation algorithm and hence a node can sleep and wake up according to information made available by message communication.

The work in [Saravanakumar et al., 2010] introduced the concept of Sleep a wake node schedule in the Low-Energy Adaptive Clustering Hierarchy (LEACH) protocol to address the problem of energy efficiency for both homogenous and Heterogenous types of WSNs. In particular, using the benefits of the LEACH Cycle in which node scheduling and Re-clustering head rotation mechanism by rotating cluster head positions to distribute the energy nodes among all nodes in the WSNs. Extending the LEACH properties for a Homogonous type of network. The run-time scheduling is a problem for Large-scale networks due to the low channel capacity which would lead to the condition. There will be loads of control packets which will food the network which will affect network quality. Unlike our work, the sleep scheduling is in the design time where the generation of control packet is very low hence providing a good QoS.

An algorithm called Lightweight Integrated Protocol Suite (LIPS) was proposed by [Tate and Bate, 2011]. The algorithm assumes that sensor nodes within a cell can hear each other and hence synchronize. In WSNs, the wireless links are subject to many sources of obstacles which makes it difficult to maintain connectivity. In a forest fire system scenario, as nodes are deployed in a harsh environment and surrounded by dynamic obstacles, this would have an impact on energy consumption and the time a WSN takes to converge to the ready stable state.

Moreover, [Yang, 2011] addressed a sleep scheduling strategy at the application layer where a set of sensor nodes is selected and activated according to a cost function to extend the network lifetime of the WSN. Although the proposed selection strategy based on the Sensor Usage Index (SUI) seems to have achieved a near-optimal lifetime, the periodic broadcast of a beacon control message by the base station every half an hour to manage the WSN would lead to communication overhead which in turn leads to energy inefficiency.

An interesting approach proposed by [Chen et al., 2014] where a sleep-awake cluster-based routing algorithm is introduced to address the MOO problem. The basic idea is to group nodes in the same neighbourhood into clusters where only one node is awake at a time within each cluster to collect and send data to the base station. [Chen et al., 2014] address the MOO problem by introducing a clutter head selection mechanism that alternates between sleep and awake mode based on the node's residual energy level. Thereby, extending the network lifetime and maintaining connectivity and coverage. [Chen et al., 2014] proposed a cluster-based routing Scheduling to be applied over the WSN which already has a sleep-awake schedule; the sleep-awake schedule has to be augmented with the routing protocol implemented. However, this approach does not discuss synchronisation between the routing and sleep-awake schedule mechanisms, which in turn will generate more control packets and will introduce traffic congestion in the network as a result there will be energy loss.

Similarly [Mostafaei et al., 2017] proposed a Sleep Scheduling Algorithm called Partial Coverage with Learning Automata (PCLA) for managing the Sleep Awake schedule in WSNs. The main goal of the PCLA algorithm is to optimise energy consumption to extend the lifetime of the WSN. [Mostafaei et al., 2017] consider the partial coverage problem in WSNs, where only a portion of the target area needs continuous monitoring. PCLA utilizes Learning Automata (LA) to schedule sensors into sleep states, aiming to minimize the number of active sensors while preserving network connectivity and coverage. The PCLA algorithm operates in two phases: (1) learning and (2) partial coverage. Initially, a backbone set of nodes is selected to ensure connectivity, and then additional nodes are activated if necessary to meet coverage requirements. The authors provide a detailed theoretical framework for PCLA, including formal definitions, analytical models, and complexity analysis. [Mostafaei et al., 2017] demonstrate that PCLA has a lower time complexity compared to state-of-the-art scheduling algorithms and that it performs better in terms of the ON nodes ratio and network lifetime. The effectiveness of PCLA is validated through simulations, showing its superiority over existing methods in various node distribution schemes, such as, (100×100, 200×200) [Mostafaei et al., 2017] includes a comprehensive performance evaluation, comparing PCLA to other algorithms under different conditions, such as network resources and coverage requirements. The results indicate that PCLA is scalable and efficient, particularly in dense networks with higher coverage requirements. The study by [Mostafaei et al., 2017] presents PCLA as an energy-efficient and scalable solution for managing partial coverage in WSNs, capable of prolonging network lifetime and optimizing resource usage. The authors suggest that PCLA's performance enhancements are significant, especially in scenarios with stricter coverage constraints and larger network sizes. In other words, the sensing range is always fixed.

Additionally, [Srivastava and Hasan, 2018] proposed a selective sleep-wake node scheduling algorithm based on collected sensory data that the base station shares with the cloud. The main goal of introducing cloud computing is to make an informed scheduling decision and overcome the limitation of a resource constrained WSN, mainly energy. The main idea is to synchronise the variations of sleep awake schedule of inter sensors and extend the overall network lifetime in WSN. The proposed algorithm uses a flag-based scheme where the cloud periodically checks the sensor data from the sensor nodes and compares it with a set of rules to put sensor nodes to sleep. Unlike the [Srivastava and Hasan, 2018] algorithm where the main consideration is energy only, this thesis addresses the intra-sensor nodes where network lifetimes, connectivity, and coverage are considered. In addition, [Srivastava and Hasan, 2018] used a simple WSN where nodes are always ON to compare their approach.

Furthermore, [Shagari et al., 2020] presented the Traffic and Energy Aware Sleep-Awake Cluster-Based Routing algorithm (TEAR), in which nodes sleep and awake based on their traffic rate and energy. The idea of TEAR is that a node with a high terrific rate of low energy is not selected as a cluster head, while nodes with a low traffic rate and high energy are selected as a cluster head. TEAR does improve the network lifetime, but it suffers from message overhead as its mechanism depends on sending messages continuously for the election of the cluster head. In addition, TEAR suffers from redundant message exchange that dissipates energy unnecessarily which in turn impacts data efficiency. To solve this problem, the authors [Shagari et al., 2020] proposed a hybrid approach, Energy and Traffic Aware Sleep-Awake (ETASA) introduced the sleep-awake mechanism by pairing nodes that are within a cluster where nodes sleep and awake to communicate with the cluster head. In ETASA, a revised version of Sleep-awake Energy Efficient Distributed (SEED) is introduced in which one slot is provided for each cluster. Consequently, improves the load balancing where energy is consumed efficiently. In-network processing is an approach to save energy; however, it is at the expense of message communication which causes overhead and delays. In this thesis, the goal is to explore the state space to find an optimal solution.

The author in [Vashisht, 2023] proposed a sleep-a-wake schedule algorithm in which the timing of the sleep and waking is determined. The proposed solution consists of two components where the first component is asleep-awake algorithm to put some nodes to sleep and the second component is a wake-up algorithm to wake up the sleeping nodes. The author controls the sleep-wake scheduling using parameters such as the node's energy level, the acceptable transmission rate, the current network load, and the network topology. [Vashisht, 2023] proposed a predefined scheduling algorithm to systematically study the variation of residual energy that is accumulated over different scenarios over the network parameters.

Moreover, [Vashisht, 2023]compared his work against the four scheduling algorithms and found significant improvement in the efficiency of various network services. The author in [Vashisht, 2023] did not use hybrid scheduling techniques that utilise clustered nodes in the network. Instead, they used one technique for the whole network. This work focuses on only the sleep-awake module of the sensor nodes network, without considering the performances of other modules We have holistically covered the net efficiency of the network considering all modules viz network module, sleep awake, connectivity coverage, energy considerations residual, and spent energy.

The authors of [Rudati et al., 2023] proposed to extend the lifetime of the WSN by combining the power of the transmission and the periodic transmission time. The basic premise is that by varying the transmission Power Level between the sensor nodes and the base station, the optimum transmission Power Level a sensor node needs is obtained. Combining the optimum transmission (Tx) Power Level and periodic transmission time, the authors found that the power consumption decreased by about 42% and the power supply's lifetime increased by about 71% in the 280m distance between the sensor node and base station, with a 108 Wh power supply. As a result, the optimum periodic transmission time is 8 seconds, and the power supply lifetime is 40 times longer. The goal of [Rudati et al., 2023] work focuses only on increasing the network lifetime of the network specifically by adjusting the signal strength needed for sending and receiving. This research considers other important objectives: connectivity and coverage where the goal is to improve the overall dependability of the WSN which is beneficial in many application scenarios such as, advanced fire detection system, intrusion detection system.

Another comparable study in [Pandiyaraju et al., 2023] proposed a multi-objective clustering approach combined with deep learning to enhance energy efficiency in WSNs for agricultural applications. [Pandiyaraju et al., 2023] introduced a hybrid optimization technique called Election-based Aquila Optimizer (EAO) to select the best Cluster Head (CH) integrated with a neural network to improve clustering precision and training accuracy. utilize a multi-objective function neural network taking into consideration parameters such as energy consumption level, the distance of the sensor nodes to the base station, the inter and intra cluster distance among sensor nodes, and the communication to form the clustering routing WSN. In [Pandiyaraju et al., 2023] the proposed precision agriculture base WSN performed well where it shows superior performance over other approaches, with classification accuracy of 99.23%, Throughput of 76.92%, Packet Delivery Ratio (PDR) of 99%, network lifetime of 98.24%, and maximum energy consumption of 50%. In [Pandiyaraju et al., 2023] connectivity and coverage are ensured when all CH is optimally selected. The work can be improved by introducing a lightweight routing mechanism that can optimise the routing path further. In addition, a lightweight neural network reduces the learning rate so that a decision can be made in a short time.

In [Bajaber, 2023] a node scheduling and partial coverage algorithm is presented for WSNs (NSPC) to improve energy efficiency and network lifetime in wireless sensor networks. The NSPC uses a scheduling phase and a sensing phase to coordinate the activities of the nodes, where sensor nodes are divided into three states: active, active redundancy (for example acting as a sleep state in terms of consuming energy), and standby. The NSPC algorithm adopts eligibility rules to determine the active and standby nodes, where the active redundancy node is put to sleep. The active nodes are responsible for sensing and data transmission, while the active-redundancy nodes alternate between active and standby modes to conserve energy. The standby nodes are those whose coverage areas are already covered by other nodes, so they can be put into sleep mode. The simulation results show that the NSPC algorithm can significantly improve the network's stability period and overall lifetime compared to the LEACH protocol, especially as the number of deployed nodes increases. The limitations of the NSPC algorithm are: (1) The states can be further divided into more meaningful states such as (active-standby, transmit-active, receive-active and sleep) which means that this algorithm has room for improvement in state optimisation, (2) This algorithm does not employ Time Division Multiple Access (TDMA) so the channels of communication are limited or fixed from the beginning, and (3) The proper optimal data transmission is possible if the state is properly optimised according to the networking characteristics such as clustering etc.

Alternatively [Harizan et al., 2024] presented a multi-objective optimization problem for the placement of Relay Nodes (RNs), Relay Nodes are powered access points acting as sub-base stations, in a WSN to ensure coverage of all deployed sensor nodes, connectivity among the RNs and the base station in WSNs. The main idea is to minimize the Intra-Cluster Distance (ICD), where energy consumption is optimised to extend the network lifetime. To achieve this goal [Harizan et al., 2024] introduced four evolutionary algorithms, namely, PSO, Differential Evolution (DE), Multi-Objective Differential Evolution (MODE), and Multi-Objective Particle Swarm Optimization (MOPSO) - to solve the coverage and connective problem in WSNs. The solution vectors are efficiently encoded, and appropriate objective functions are derived for each algorithm. In [Harizan et al., 2024], the justification for using such Relay Nodes will provide better coverage connectivity and lifetime. However, using the above four evolutionary algorithms collaboratively in such resource constraints in WSN will introduce computational complexity.

Furthermore, in [Hameed and Idrees, 2024] an Energy-Aware Scheduling Protocol (ESP) based on a hybrid metaheuristic technique is proposed to optimise the lifetime of WSNs. The ESP protocol is composed of two main phases. (1) A distributed clustering phase, where sensor nodes are grouped into clusters is to be monitored. A periodic cluster head election approach was utilized according to the residual energy of the nodes, the number of neighbours, and the distance for each node in the cluster. (2) Sensor node scheduling phase, In this phase, the scheduling optimization model is constructed with three objectives, (a) reducing the number of uncovered zones in the area to be monitored, (b) minimizing the number of active sensors, and (c) selecting the active sensor nodes with the maximum residual energy. [Hameed and Idrees, 2024] solved the MOO using a hybrid metaheuristic algorithm that combines the Cuckoo and Genetic algorithms. The problem of these types of considerations does not hold, as there is no logic why cuckoo should be coupled with a genetic algorithm. The coverage is attained sector-wise i.e., targeted coverage which has its limitations as some nodes will be in a sleep mode and there will be coverage holes created apparently.

In alternative work, [Jaiswal and Anand, 2024] address connectivity, coverage and lifetime in homogenous network scenarios, the author did not consider heterogeneity. Until for a specific purpose the WSN coverage should not be target limited based. As better coverage provides better utilisation of the network. In [Harizan et al., 2024], the justification of using such Relay Nodes will provide better coverage connectivity and lifetime. However, using the above four evolutionary algorithms collaboratively in such resource constraints in WSN will introduce computational complexity. The authors in [Shagari et al., 2020]proposed a hybrid approach, heterogenous Energy and Traffic Aware Sleep-Awake (ETASA) introduced the sleep-awake mechanism by pairing nodes that are within a cluster where nodes sleep and awake to communicate with the cluster head. In ETASA, a revised version of SEED is introduced in which one slot is provided for each cluster. Consequently, improves the load balancing where energy is consumed efficiently. In-network processing is an approach to save energy; however, it is at the expense of message communication which causes overhead and network delays. In [Pandiyaraju et al., 2023]connectivity and coverage are ensured when all CH is optimally selected. The work can be improved by introducing a lightweight routing mechanism which can optimise the routing path further. In addition, a lightweight neural network reduces the learning rate so that a decision can be made in a short time.

The author would like to draw the reader’s attention that the deployed machine learning methods in the WSN in this thesis are not invented from scratch, they were used in past WSN literature, however in a different context addressing different problems such as routing, load balancing and security problems. These methods were deployed specifically in the scheduling approach for example at the application layer of the OSI stack of the WSNs. Below is the related work in each component of this thesis that has been deployed and differentiated from the work presented in this thesis as per component.

2.2 A Perceptron Multi-Layer Neural Network-based Approaches

There is a wide range of related works on the application of the ANN model for solving a different problem in WSNs, the interested reader may visit the surveys in [Aliyu et al., 2019] and [Vidhya, 2023]. For example, the authors in [Alrajeh and Lloret, 2013] utilised ANN for intrusion detection systems to address security aspects such as integrity, confidentiality and reliability of WSNs. Other work proposed by [Zroug et al., 2023] makes use of ANN to evaluate the performance of WSNs. Other related works which considered the use of ANN can be found in [Sharma et al., 2015], [Wan et al., 2018] and [Masti et al., 2019].

The authors in [Sharma et al., 2015] proposed to enhance the network lifetime by the use of ANN in the Cluster Head selection mechanism in WSNs. Sharma et al, 2015 consider residual energy for selecting the Cluster Head nodes. The radial basis function network model is used for the cluster head selection mechanism. The new enhanced LEACH-C protocol is compared to the original LEACH and LEACH-C protocol where the result is to be found that the new enhanced version improved in terms of energy utilisation. Unlike the approach proposed by [Sharma et al., 2015] where the work is targeting the routing protocol at the network layer, the ANN analyser is used in this thesis to analyse the complex behaviour of WSNs, such as the continues and unpredictable dynamic changes of the network topology due to energy constrains of the sensor nodes.

[Wan et al., 2018] proposed an Energy-Efficient Sleep Scheduling algorithm (ESSM) using the fuzzy logic principle with similarity measures for WSNs. The basic idea of ESSM is to turn on a set of sensor nodes in the target area where it is highly dense. Hence, exploiting the nodes in dense areas and reserving a level of energy. The approach has limitations in that it is sensitive to parameters initializing which will lead to overlapping of the selection of the set of nodes that have similarities. In this solution, the fuzzy mechanism depends on nodes that are densely deployed in the network. Hence, it becomes difficult to evaluate the performance of the algorithm in less dense areas or where nodes are spars, especially in large WSNs.

The authors [Masti et al., 2019] attempted to address the problem of reducing computation and memory capacity in virtual sensor scheduling (Software that is embedded in an IoT device that acts like a sensor). The author [Masti et al., 2019] utilises the use of ANN to discover patterns that are of use for WSNs activities such as node scheduling, energy consumption, packet communications etc, In particular, through the use of linear observers where the processed dataset and the inputs and outputs measurements where time-variant model are fed to delineate and obtain the abstracted data to fit the virtual sensor. Since there is an architecture difference, [Masti et al., 2019] try to delineate the data to fit the architecture of virtual sensors. Hence, enables to mapping of the data into the virtual sensors. The ANN-based scheduling algorithm proposed in this thesis is built for a real sensor node while keeping in mind the concept of longer lifetime, coverage and connectivity simultaneously whereas the solution of [Masti et al., 2019] virtual sensor nodes design irrespective of energy, coverage and connectivity considerations hence, they are called Virtual sensors.

2.3 Hidden Markov Decision Process on Approaches

There are several papers in the literature that used the HMM model to address multiple problems that are related to energy consumption in the WSNs [Hu et al., 2007]. Therefore, renderer the use of the HMM model in WSNs is not new. What follows sheds light on some of the related work that utilizes the HMM model in WSNs.

The use of the HMM method in WSNs was presented by [Krishnamurthy, 2002], where a scheduling algorithm that utilized the HMM model to observe states for a resource-constrained WSN was proposed. The work addresses the problem of data reliability of obtaining certain physical data e.g., measurements of certain processes, known as signal processing applications. The basic principle is to utilize the HMM finite states to locate noisy sensor nodes in the WSN and then select these nodes to send the required data. The solution utilized a dynamic stochastic programming which is achieved in two folds: (1) find the optimum channel allocation among various components of a measurement vector. For example, transmitting within a time-shared communication channel but with limited noise sensor bandwidth. (2) given resource constraints, best to identify optimal measurement periods of the sensor while exploring the trade-offs between rate and measurement ranges. Identifying the optimal sensor measurement schedule is a scheduling problem manifested by dynamic programming. However, Bellman's equation of dynamic programming does not necessarily lead to practical solutions [Krishnamurthy, 2002]. The reason for this is due to the cost of dynamic programming per recursion over innumerate infinite set. Their paper includes optimal dynamic programming-based HMM sensor scheduling algorithms which take into consideration the WSNs constraints

- including sensor management, estimation, and utilisation constraints. Steady-state HMM scheduling, however, is presented for larger-sized WSNs where the computational complexity of dynamic programming is restrictive.

[Goudarzi et al., 2010] proposed the use of HMM in combination with Particle Swarm Optimisation (PSO) to predict the energy level in WSN. The algorithm used PSO to select the cluster head Nodes. The proposed methods reduce the cost of clustering and improve the performance of the WSN. The optimization problem defined for the best possible energy values can be obtained by solving it with PSO to find the best position that evaluates from the fitness function. To initialize the PSO all the defined variables are assigned with random values in the search space. In each run, each particle generates the best individual position (associated with energy) and global best position to obtain the best possible solution in search space. The communication overhead is reduced due to the use of PSO. In comparison, this thesis's work in reducing energy utilized a very simple HMM algorithm where the energy was optimised as well as the position of nodes for effective scheduling and energy management. Meanwhile, the work by [Goudarzi et al., 2010], involves complexity in producing the intended solution. In such work, there is always a trade-off between the cost of finding the energy versus the accuracy. If the cost is higher the accuracy is better and a lower cost leads to lower accuracy. This solution introduces an overhead in the WSN because of the complexity involved in using the PSO algorithm at the expense of opening a good level of accurate energy solution.

[Qihua et al., 2015] proposed a scheduling algorithm to extend the network lifetime. HMM is utilized to model sensor node states in the WSN where a node can be in two states 0 asleep and 1 active. The approach bases its HMM model on two factors that are the energy cost and the errors of the sensor nodes. In addition, this approach considers the use of an actuator which takes the reading of the node's energy and sends that data to the central controller which manages the entire WSN. This approach seems to improve the lifetime of the WSN however, the introduction coordinator led to an overhead control message on the WSN. The approach adopted in our thesis uses a simple method of utilizing the HMM model to extend the network lifetime and has no control message overhead.

In [Bhatia et al., 2019] proposed Prediction Based CH Selection (PBCS) algorithm that combines PSO for clustering and an HMM for selecting CHs based on predicted energy levels of nodes. The basic idea is to dynamically cluster the WSN to avoid node failures this will maintain connectivity and implore network lifetime. The authors presented their work in two parts one is clustering and other is cluster head selection both are performed with two computationally intensive algorithms like PSO and HMM chains. The outcome is measured by using some conventional routing algorithm like flooding and spanning tree protocols to check the energy considerations. But this paper does not discuss about network lifetime coverage which is the essence of the WSN communications.

In summary, to the author's best knowledge, there are not many works that utilized HMM to address multi-objective optimization problems involving energy consumption, network lifetime, coverage, and connectivity, in the context of the WSNs scheduling approach. In comparison, our work in reducing energy utilized a very simple HMM algorithm that we used to optimize the energy as well as the position of nodes for effective scheduling and energy management. Hence, our approach utilizes the HMM method to extend the network lifetime and introduce no control message overhead.

2.4 Bio-inspired Approaches

Bat-inspired algorithms (part of NIA's) have been used by the scientific community for solving hard optimisation problems, as they have been extensively surveyed in the [Chawla and Duhan, 2015].Bat algorithms solve optimization problems in complex systems such as WSNs, specifically, by utilizing a cost function to find the best possible value of the solution in the search space [Sendra et al., 2015]. There are several bioinspired computation approaches for optimizing energy only such as Genetic algorithm [Jin et al., 2003] and [Huang et al., 2010]. Other bio-inspired flocking techniques were used to detect anomalies in the stream of data in the WSNs [Forestiero et al., 2009]. The interested reader may refer to the comprehensive survey of node scheduling schemes in WSN.

In the paper [Forestiero et al., 2009], the authors solve the problem of topology management in WSNs. First, the solution is restricted to 2D space. Second, they are trying to use the density distribution of the sensor nodes in the network. In particular, the authors are using the flocking model in which the agent is moving into a space in a fixed time. When these agents encounter another agent, they will form a flock if they are similar in density. Since these flocks form a swarm so the offline computation can easily be avoided using this approach. The approach assumes the agents are on when sensor nodes are active which is not the case in the WSNs as sensor nodes alternate between ON and OFF rendering the density changeable.

In [Nilsaz Dezfuli and Barati, 2019] the author proposed a Distributed energy-efficient algorithm for ensuring coverage of wireless sensor networks for addressing a MOO problem in WSN. The main idea is the application of Firefly to grid-based computing used for wine. Unlike bat optimisation techniques, the firefly when in a higher energy state flies at a higher altitude when the energy is low it settles down or on leaves etc. Taking several fireflies in this case, a part will fly another will rest, which creates an equilibrium or optimization sense when applied to WSN. Hence, in [Nilsaz Dezfuli and Barati, 2019] the method involves dividing the network area into square cells and selecting the node with the highest energy in each cell as an aggregator. The firefly algorithm is then used to choose and maintain the most suitable node in each cell, maximizing coverage while minimizing energy demand and extending the network lifetime.

The research work was proposed by a Bat-NIA algorithm called target-connected coverage. The author formulated the problem by proposing a couple of bats that are based on a binary search space. The first bat's main goal is to find the active node in the WSN, and the second bat's goal is to use the active node to find at least one path to the sink node to report data maintaining connectivity. The bat algorithm proposed in this paper is based on a continuous search space, not a binary search space as proposed. This is because energy varies with respect to time i.e., as time progresses energy is depleted. Furthermore, the algorithm has approached the same problem but from a different perspective as the authors did not consider node scheduling. Nodes were assumed to be always in one state e.g., active state. However, the work presented by [Kim and Yoo, 2020] provides a sleep-awake schedule for each node. Introducing node scheduling has increased the complexity of the multi-objective optimization problem in this thesis, e.g., working with more than states of nodes sleep and wake. Additionally, the mechanism proposed by [Kim and Yoo, 2020] did not provide a sleep schedule for each node in the network but limited the proposed algorithm's search space to find the active nodes only and always maintain a link to the sink.

The authors in [Narayan et al., 2023], addressed the MOO problem by introducing the clustering approach in which fuzzy logic is utilised. The main idea behind this work is to construct a fitness function in which swarms work collaboratively to optimise a fitness function and send data from all sensor nodes to the Base station. [Narayan et al., 2023] attempted to reduce the residual energy of each sensor node in which the minimum amount of energy is consumed. Using a fuzzy controller can ensure that effective routing is done considering the above condition as the backbone. Our approach for the Bat Node scheduling algorithm is similar to the approach of [Narayan et al., 2023] which has MOO using PSO. The proposed BAT approach will have the same formulation for the MOO problem, but the involved parameters are different.

An interesting example proposed by [Yang and Xia, 2024] a cuckoo search based on Cauchy optimization in which it addresses both coverage and routing in WSNs. The main idea is to use a dynamic cluster radius for routing based on a non-uniform clustering type of WSN. [Yang and Xia, 2024] addressed the coverage and routing challenges in the network but did not address the connectivity issues, and node scheduling i.e., in confirmation with TDMA which is practical while implementing in real life. The results indicate that the proposed algorithms outperform traditional methods in terms of coverage, energy efficiency, and network stability. However, the delivery rate is not considered, [Yang and Xia, 2024] present a very unrealistic view of energy loss of less than 1% which is the case if fewer packets are delivered and almost no control packets are generated in this case. The research suggests that while the proposed algorithms show promise, there is room for further exploration of the complex structure of WSNs.

In [Ovabor and Atkison, 2024] a hybrid Multi-objective Optimization technique integrates cascading firefly and Greywolf algorithms to improve connectivity, coverage and network lifetime in WSNs. [Ovabor and Atkison, 2024] coupled with the Pareto front in which the optimal solution provides insights on the trade-off between the three optimised objectives. The algorithms complement one another in terms of their strength. For example, firefly's optimization techniques are better for connectivity and Greywolf's optimization techniques are better for coverage, now in this case if they are applied in a cascading manor, they will complement one another resulting in better connectivity and coverage. The difficulty of this type of hybrid approach is in the increase of computational complexity and more convergence time required for optimization.

Another example that addressed the MOO problem is proposed by [Jaiswal and Anand, 2024] in which a node deployment algorithm using Firefly Optimization (FA) to achieve target location coverage and sensor node connectivity is introduced. To ensure the best QoS [Jaiswal and Anand, 2024] introduced a multi-objective fitness function that considers factors like maximum survival, distance from the base station, coverage, number of sensor nodes, and connectivity. Furthermore, [Jaiswal and Anand, 2024] introduced the distance to the base station as an important factor for delay-sensitive applications. [Jaiswal and Anand, 2024] achieved fault tolerance through the p-coverage and q-connectivity which make the network perform efficiently in the complex. Although [Jaiswal and Anand, 2024] address connectivity, coverage and lifetime in homogenous network scenarios, the authors did not consider heterogeneous which is the common deployment scenario in WSNs. Until for a specific purpose the WSN coverage should not be limited to a target-based coverage, as better coverage means better utilization for WSNs.

Moreover, the authors in [Jia and Zhang, 2024] proposed to improve the existing Graywolf algorithm optimization techniques and provide an energy saving coverage model. The model ensures that at least each point in the target area is covered by at least one sensor node. This work is an improvement over the existing Grey wolf algorithm used in WSN, it has significant improvement over the original which provides better coverage and energy saving in the process.

In the literature, Bat algorithms are used via their prey-hunt process [Sendra et al., 2015]. However, the novelty of the proposed work in this paper is manifested by using a new concept of the Bat algorithms in which bats' wake-sleep process is used for the multi-objective optimization problem which this thesis solves. In particular, the Bat theory was modified based on experimental results.

2.5 Self-Organising Map Approaches

There are several approaches in the literature that use SOM to address the problem of energy optimisation in the WSN [Miljković, 2017]. For example, the author in [Cordina and Debono, 2008] utilised SOM to propose an energy optimisation model to reduce communication in clustered WSNs. In particular, the author proposed an approach for cluster selection based on the Extension to Growing Self-Organising Map (E-GSOM) technique. The author compared two evolutionary-based genetic computing techniques against the E-GSOM where the result showed an improvement of energy efficiency among its peers.

Another work that utilised SOFM to provide efficient cluster routing algorithms was proposed by [Cordina and Debono, 2008]. The work reduces energy consumption by organizing sensor nodes into clusters whereby better communication nodes in the WSN are minimized. The solution improves the network lifetime via the distance reduction among CH nodes and load balancing cost function. Since communication is the most expensive energy expenditure in WSN, the proposed solution managed to optimize communication distance among CHs, reducing the closest communication distance to the base station. Although this approach extends network lifetime, by reducing the communication distance to the base station. The proposed approach incurs messages overhead due to the setup cost of forming the CHs with the short distinct to the base station.

On the other hand, the author in [Bai et al., 2008] introduced the Self-Organising Map Scheduling Algorithm (SOMSA) for sensor networks where the scheduling of the nodes is based on dynamic environment settings with local interactions. These settings are entirely based on the changing energy levels of the WSN. The SOM, therefore, uses these patterns as input vectors to produce dynamic scheduling for efficient data transfer. In our case, we already computed the final scheduling before the WSN operation, but in [Bai et al., 2008] case, the scheduling is computed using the SOFM dynamically, hence the energy consumption is more in this case than ours due to the message control exchange overhead.

In [Patra et al., 2010], the author proposed the SOFM Topology Building (SOFMTB) Algorithm to optimise energy in WSNs. The work utilised the SOFM technique to adjust the clusters based on energy levels for efficient dissemination of data over the network. The SOFMTB divides the WSN into tiers using the k-means algorithm as cluster head and cluster members. It also modifies the topology of the network providing efficient partitions in the network.

In the work of [Mittal and Kumar, 2015] SOFM was used to reduce energy consumption and bandwidth usage in a resource-constrained WSN. The solution tends to reduce the data communication during the entire network lifecycle by reducing its size. For instance, in this work,1500 data volumes were generated and aggregated from all sensor nodes and sent to the base station. This approach clusters these data and reduces its size by selecting the most suitable size that has a meaningful one and sending it instead of that 1500 large volume one.

In [Kulkarni et al., 2021] a Dynamic Decision-Making Algorithm (DDMA) is proposed for sensor node deployment in a heterogeneous WSNs using SOM model. The DDMA algorithm has two phases - a sensor setup phase and a sensing phase. In the sensor setup phase, the algorithm uses SOM to pair each coverage point in the deployment area with a nearby sensor node. This ensures that all coverage points are monitored by at least one sensor. In the sensing phase, when a sensor node fails, the algorithm uses the pairing information to identify the uncovered coverage point and instructs a neighbouring node to move and cover it. The main objective of this work is to address the coverage hole problem by introducing node mobility. The solution to the problem is practically not feasible as in remote environments the nodes cannot be moved through locations unless it is mounted on some vehicle.

On the other hand, the work in [Talei et al., 2023] used the self-organising map technique for high buildings, in the HVAC system, to find energy-saving opportunities and thus distribute the energy efficiently. This work used one year of data set in time serious analysis to provide service that accommodates user requests regarding outdoor/indoor temperature. Utilising SOM has proven to provide energy energy-efficient solution.
In summary, having assessed the latest state-of-the-art SOM technique in WSN, it has been noticed that there is still more room to state space exploration and obtain the best optimal solution to find the optimal scheduling for the MOO problem.

2.6 Long Short-Term Memory Approaches

Several studies in the literature have looked into time series prediction using the Long Short term Memory approach, LSTM, to tackle various issues in WSN, including energy efficiency, as demonstrated in the works by [Mohanty et al., 2020; Shu et al., 2019]. In their work, [Shu et al., 2019] suggested optimising energy consumption by means of minimising the number of communication links between the Fusion centre/Base station (BS) and the sensor nodes in WSNs. [Shu et al., 2019] utilised a dual prediction scheme (DPS) based on a Least Mean Square (LMS) filter to predict the number of communications links between BS and the sensor nodes. The sensor nodes can only send data when the error margin between the sensed data and the predicted data is below a certain user-defined threshold. A Beacon signal is then used in the network to notify nodes (sender/Receiver) to use the predicted value of LSTM as the current time step to send the actual data. The use of Beacon single consumes less energy in comparison to sending large data packets. The limitation of this approach is that it considers stationary nodes that cannot be employed with the dynamic nature of the WSNs. In addition, [Shu et al., 2019] consider no node failure scenario that is unrealistic for WSN.

In [Cheng et al., 2019] a Multi-Node Multi-Feature (MNMF) model based on bidirectional LSTM networks is presented. The main aim is to reduce unnecessary data transmission thereby improving the quality of the data collected in WSNs. To exploit spatial-temporal correlations among sensory data. To address this, the MNMF model first enhances data quality through the quartile method and wavelet threshold denoising. First, quartile and wavelet threshold methods were used to filter and remove noise from the data to improve its quality. After denoising the pre-processed data feature extraction is performed with bidirectional LSTM. Finally, the merged layer of the neural network (i.e., the last layer of LSTM) fuses these pre-processed data features to predict future sensory data. This approach utilises LSTM for sensory data prediction only, this algorithm is not at all catered towards implementing the connectivity, coverage, and a lifetime of the network, on the contrary, our work focuses on predicting the data as well as catering towards the best coverage, connectivity, and lifetime of the WSN.
The author in [Mohanty et al., 2020] attempted to reduce energy consumption by utilising RNN-LSTM based model. The used LSTM based components split the WSN into a different layer on the sensor nodes. The approach by [Mohanty et al., 2020] reduced data communication by utilising LSTM in that it compared the actual data and the data that needed to be sent. Thus, reducing the communication and balancing the workload in the sink node. The use of the LSTM in the [Mohanty et al., 2020] approach is limited to communication reduction only. Unfortunately, great details were spent on reducing communication, yet no efforts were spent on coverage.

A more recent work proposed by [Jeyalaksshmi and Ganesh, 2022] utilised the Bidirectional Long Short-Term Memory (BiLSTM) technique and provided adaptive duty-cycle scheduling for WSNs. The BiLSTM stores the previous and future steps to adjust the node's sleep schedule. The algorithm employs an eligibility check based on the Jaccard Similarity Index (JSI) value among sensor nodes. The nodes that have a higher value are elected to sleep where their sleep schedule is extended based on data patterns obtained (e.g., traffic load and energy level) by BiLSTM. This work assumes that sensor nodes are already in a sleep mode where the intervention is on their sleep duty cycle only. The BiLSTM algorithm did not consider network coverage and connectivity as it assumes a random duty cycle for all the nodes. Also, as nodes progress in time the legibility rule will take its toll on the residual energy hence on network performance.

Other works such as [Salmi and Oughdir, 2022] have used the LSTM model to detect Distributed Denial Of Services attacks (DDOS) in WSNs. Our work, however, utilised the LSTM model to address the MOO problem in WSN from a different and new perspective i.e., to use data centric modelling which is to detect a certain useful sequence of events (data) that has a good degree of coverage performance, connectivity, and a lifetime to predict a useful LSTM based node scheduling model.

2.7 Summary and Concluding Remarks

The literature itself shows that the MOO problem in WSNs is not trivial [Purushothaman and Nagarajan, 2021], especially when node scheduling is the solution. Most of the reviewed works lacked focus on the dependability issues of WSNs including coverage, connectivity, and lifetime of WSN nodes [Thomas et al., 2020]. The problem exhibits more complications when the considered type of the WSN is for safety-critical applications. This chapter has reviewed state-of-the-art node scheduling ANN, HMM, Bioinspired Bat, SOM, and LSTM approaches where the literature has mainly focused on the energy-efficient aspect of WSNs. In contrast to most published works in the literature that focused on optimised energy efficiency (network lifetime), the focus in this thesis includes the WSN's service reliability (nodes' connectivity and coverage) as well as service availability (network lifetime).

To the author's best knowledge, after assessing the available node scheduling approaches in the literature, a limited amount of research was found to have focused on the MOO problem of the WSN node scheduling that focuses on network lifetime and coverage. For example, [Cardei et al., 2005] consider WSN coverage requirements, but without ensuring a good level of network connectivity which hinders service reliability and availability at some point in time. The Authors in [Tian and Georganas, 2002] didn't cater for network coverage instead they assumed a simple scenario of no coverage.

In addition, no work addresses the network's service reliability together with its service available. The RCS [Liu et al., 2006] and [Wang et al., 2010] scheduling algorithms were found to focus on the aforementioned three optimisation objectives. Hence, the RCS algorithm is the earlier to address the MOO problem **(cf. Section 1.2).** and easy to replicate it was selected as the basis of the work in the upcoming chapter. The argument is that WSNs sometimes operate with such resources in a harsh environment where there is no human intervention. Hence, the level of complexity in predicting the performance of WSNs during the run time is extremely complex. This necessitates the use of a technique such as ANN that has the ability of dimensionality reduction to help in reducing such level of complexity particularly with predicting the lifetime of each sensor node. Consequently, it is reasonable to use the ANN to simulate and analyse the RCS performance.

Limitations of the state space exploration: The proposed ANN analyser, train, analyse and detects the Node's sleep and wake cycle Hence, helps in pointing out limitations of the RCS-based scheduling algorithm e.g., it provides a classification of the overused and the underused Nodes. Thereby, conclude more room for state space explorations. Hence, the ANN analyser advantage points out in limitations of the algorithm itself. Unlike, traditional simulation compares algorithms against one another in terms of performance, but it does not tell why they are not performing as they should, the ANN analyser can help to find if the algorithm is optimal or not.

Given the limitations of the RCS algorithm, it would be possible to extend the network lifetime e.g., by making use of the nodes which are in the 2-Hop area joining the 1-Hop area and balancing their duty cycle in terms of communication loads. The ANN analyser confirms that the RCS algorithm is not good enough in terms of state space exploration, so we can explore the state space by the output of the ANN analyser.

Although the ANN analyser helps in analysing and identifying the limitations of the RCS algorithm, it is difficult to tune up for other algorithms. The ANN might not need to be used on the proposed algorithms such as HMM-based scheduling algorithm, Bio-inspired computation, etc. That is because those proposed algorithms are good at what they are doing, as in their search space they already proposed the optimal solution, unlike the RCS. The ANN analyser algorithm gives more room regarding the state space exploration (possible solutions). The RCS algorithm doesn't give us such information. As a result, the HMM algorithm was devised to overcome the limitations of the RCS algorithm.

Hence, the Bio-inspired Bat scheduling algorithm is assessed as a means for exploring a larger search space. As WSNs are a distributed type of network, then it is logical, that the proposed Bio-inspired Bat solution is performed in terms of distributed computing to solve the scheduling problem. Notably, this will improve and help in finding the optimal scheduling algorithms for MOO problems. Furthermore, since the aim is to solve the MOO problem and achieve the optimal solution, SOM and LSTM approaches will be used to explore different representations of the solution space. For example, SOM is used for exploring solution space spatially, Whereas LSTM is used to explore the solution in a temporal approach. In the literature, there were applications for finding optimal WSN routing paths to the Base station utilising SOM. Such approaches were used in the literature for a different use case i.e., routing or path optimality than the ones presented in this thesis.

Chapter 3 A Perceptron-Based Neural-Network Analyser to Enhance the Scheduling Performance

This chapter illustrates the MOO problem, namely the limited solution space exploration of existing algorithms, by evaluating the RCS algorithm [Liu et al., 2006]. We notably propose using an ANN-based algorithm to diagnose issues with the RCS algorithm which cannot be determined easily through simulation or by comparison with other algorithms. This case study then provides us with the motivation for exploring more systematic ways for developing efficient scheduling algorithms as a solution for critical WSN systems. Hence, this chapter starts by evaluating the simulation platform available for WSN, followed by the required parameters and the energy model. Finally set the scene for the simulation experiment using MATLAB to study the performance and behaviour of WSNs scheduling algorithm.

3.1 Simulation Platforms for WSN

There are many approaches to study, analyse and evaluate the performance of WSNs during the run time. One approach is developing a real physical hardware testbed, this enables researchers to experiment with physical nodes and analyse the data in real time for WSNs. While this approach provides accurate results, it is expensive and may not be available or affordable, which is the case with the author of this thesis.

Simulation is a widely used and accepted approach in the scientific community [Adday et al., 2024]. On the one hand, simulation is easy to use, affordable, and open-source [Zhang et al., 2024]. For example, there are many simulation environments are heavily supported in public forums such as NS-2, NS-3, COOJA and MATLAB network simulators [Horneber and Hergenröder, 2014]. On the other hand, simulations may not accurately represent the real system that the researchers intend to simulate. Nevertheless, the level of accuracy is accepted with a certain commonly used parameter value [Liu et al., 2006] and [Saravanakumar et al., 2013]. In general, an in-depth review to identify and quantify the acceptable values of parameters is required to ensure the simulation model represents a reasonable commonly used value, such parameters include the location of nodes, sensing range, communication range etc [Abdulzahra et al., 2023].

An alternative approach combines a small-scale physical testbed with simulation tools [Stehlık, 2011]. Another approach couples simulation with mathematical Theorem proofs [Timo and others, 2005]. In this thesis, well-known models of energy consumption, and communication parameters for WSNs are used. Moreover, parameter values that are accepted in the research community for sensing range, communication range, energy consumption per packet are used [Liu et al., 2006] [Bajaber, 2023] and [Saravanakumar et al., 2013]. In WSN simulations, most researchers adopted the NS-2 environment which is primarily designed for simulation routing protocol. NS-2 is not selected in this thesis because its technical support manuals are unorganized and it is very difficult to find such documentation to use NS-2 for scheduling algorithms in the field of WSNs, despite its vast documentation on routing algorithms which is not the focus of this thesis. There are no single models for WSNs, depending on the WSN area of interest in research, e.g., routing protocols, or security aspects, the WSN layer of interest has different models and simulation environments. Some researchers built their research environment to allow them flexibility in configuring the models and simulation environment [Coronato and Testa, 2013].

In terms of simulation model availability, OMNET++, JiST, and SSFNet have fewer available routing protocol models than other simulators (especially NS-2). For other simulation environments like J-Sim and Ptolemy II, the basic building blocks to create a simulation testing environment are available which can be time-efficient in building a complete model. However, issues with such solutions stem from the technical difficulty in combining building blocks to represent a complete WSN model because such solutions lack examples, tutorials and technical documentation, as they are primarily used in traditional wireless and wired networking systems, but they are not widely used in WSNs. TOSSIM, EMTOS, and ATEMU are examples of tools that can replicate genuine sensor code.

Recent simulators, such as JiST/SWAN, claim to outperform NS-2 and GloMoSim in their sequential version. Parallel simulations should, by definition, perform and scale better than sequential simulations. The cost is manifested in the increased programming complexity. GloMoSim whose goal is performance rather than scalability can simulate up to 10,000 wireless sensor nodes in parallel. The DaSSF parallel tool, which focuses on scalability, can handle network topologies with up to 100,000 connected sensor nodes.

Graphical User Interface support is included in all the aforementioned simulator environments. For animation, tracing, and debugging, OMNET++, NCTUns2.0, J-Sim, and Ptolemy provide strong Graphical User Interface frameworks. All of them include features such as parameter adjustment during execution. JiST, on the other hand, does not contain any Graphical User Interface support except for an event logger and viewer. The uncomplicated and straightforward trace reproduction Nam tool is currently supported in NS-2.

Specific tools also have remarkably detailed user interfaces. TinyViz is a Java extension to the TOSSIM visualization tool that gives important debug information. Users can create their plugins that listen to TinyViz's TOSSIM events and perform some action. EmTos has a utility called EmView, which is very similar to EmTos but written in C. Ptolemy-II and NCTUns2 both have graphical editors that are quite easy to use. Because the remainder of the packages' graphical editors are not as straightforward, it is usually better to develop models using their script-oriented approach. It is worth mentioning, that there is a definite tendency in simulations to use native code from genuine devices (e.g., TinyOS/NesC). This feature is available in all WSN frameworks. Cooja is a Java-based simulator, the programming can be extensively hard as it has very poor visualization capabilities. The software bugs are unprecedented in NS2 and NS3 because of versioning issues. Hence, it is a common problem in NS2 and NS3. In an open-source simulation tools, versioning bugs are a common issue.

MATLAB, on the other hand, is a well-documented environment with toolboxes and strong technical support and documentation for its users. Furthermore, MATLAB has the following set of advantages:

- The rapid prototyping in MATLAB gives the software engineering walkthrough with a spiral model used in developing long-term service software.

- The visualization capabilities are very crucial in displaying various performance and animated sequences of the Transmission (Tx) and Receiving (Rx) among the nodes as well as the topology geometry associated with networking between the nodes.

- MATLAB allows easy integration and reusability of the third-party plugin modules developed by many languages such as C, C++, Java, and Python. This property is inherent with MATLAB simulators which are not available with NS2.NS3.

Due to the above-mentioned advantages of the MATLAB network simulator has over other simulation tools, it is therefore selected for the work presented in this thesis.

3.2 Simulation for Scheduling algorithm in MATLAB

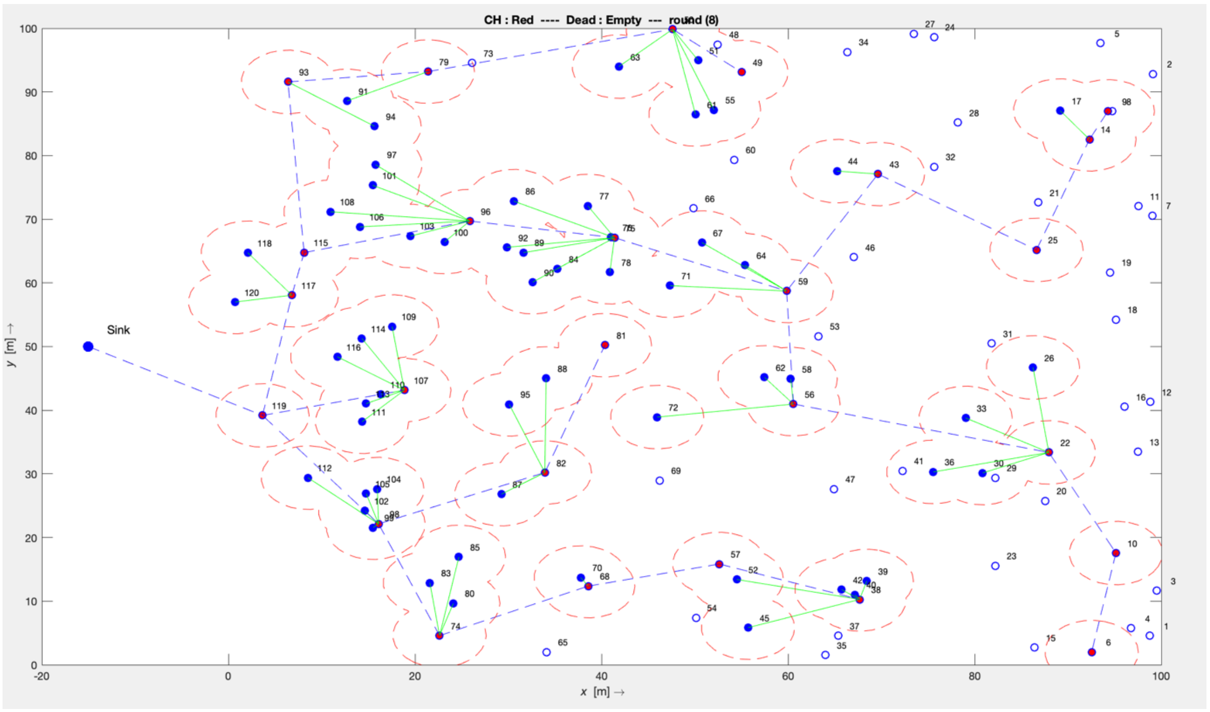


Figure 3.1 Simulation Set-up for Scheduling Algorithm

Figure 3.1 shows a WSN which consists of 1 sink node and 150 nodes, where each node has a transmission range (R) of 3.5 metres on an area of interest of x = 100, y = 100, the initial energy consumption of each node is 0.25 joules, whilst the energy required to receive and to transmit one bit is 1.0000e-09 joules, this is also the value of the data aggregation. The desired packet size of a cluster head per round is 3000 bits; all the data mentioned can be seen in table 3.1. The figure indicates how each node in the network consumes energy during the simulation run time. As the simulation runs, the network's performance can be observed where each node fully depleted energy using the standard cluster routing algorithm, without the scheduling algorithm. The result of the analysis is populated from both the classical configuration and the configuration with the scheduling algorithm's addition to compare the rate of improvement in efficiency between the two scenarios, as displayed from Figures 3.3 to 3.4. Simulation was individually tested 30 times. 1-way ANOVA analysis was implemented on the RCS algorithm to identify the p-value for statistical results.

3.2.1 Simulation Parameters

The parameters used in this work include simulation parameters, coverage, and nodes. Those parameters can be turned upwards and downwards in values which are commonly used in the literature. In this work, the same values are adopted for the simulator environment [Liu et al., 2006] [Bajaber, 2023] and [Saravanakumar et al., 2013].

Table 3.1 Parameters set within the simulation of the WSN.

Simulation Parameter		Value
Coverage Parameters	Number Of Sink Nodes	1
	Number Of Nodes	150
	Width of Y-Axis	100 meters
	Height of X-Axis	100 meters
	Distance from sink to sensor area	5 meters
Node Parameters	Initial Energy [EI]	0.25 joules
	Energy for transmitting one bit [Etrans]	1.0000e-09 joules
	Energy for receiving one bit [Erec]	1.0000e-09 joules
	Data Aggregation Energy	1.0000e-09 joules

	Energy of free space model amplifier	5.00e-08 joules
	Maximum Range of transmission per node [R]	15 meters
Scheduling Algorithm Parameters	Packet Size for Cluster Head Per Round	3000 bits
	Desired Percentage of Cluster Heads	5/100
	Packet Size for normal nodes per round	80 bits
Simulation Parameters	Number of Rounds	2000

3.2.2 Energy Model

There are multiple energy models for calculating energy consumption in WSNs, most published literature confirms that sending and receiving data packets in WSNs consume the most energy, more specifically sending packets uses the highest energy levels in WSNs.
EONi symbolises the energy consumed by node 'i' when turned 'ON' and is part of the WSN, EOFFi symbolises the similar but OFF state. When a node listens to the network and decides to send packages, this is a complicated stage. This is due to the prior state of listening to the channel before sending/receiving packets. With a free channel, the node utilises ETXi (i.e., transmission energy) to send a packet (PTXi); with a busy channel, the node generates re-try variable (NRTi) because it has to execute the CSMA/CA algorithm using the CSMA energy (ECSMAi). The node, then, will receive an acknowledgement signal, ACK, using ERXi and PRXi. In the absence of ACK reception, a retransmission PRTXi is generated. Furthermore, ESwitching energy is used for nodes' activity changes consuming switching energy. The aforementioned packet cycle includes receive, listen, transmit, and change states until network energy depletion (simulation time ends) using EOFFi. EMCi (microcontroller energy) is consumed for sampling the WSN. For calculating the energy, the method used in this work is

by multiplying time, T, with energy, E. T is the simulation period whereas E is measured in Joules. I is the current which is measured in Ampere whereas V is the voltage which is measured in Vols. The method adopted in this thesis for calculating energy consumption is by multiplying the time ‘T’ which represents the simulation period of the respective components subscripted to the variable ‘T’ with ‘E’ the energy which is measured in joules, ‘I’ is the current, which is measured in ampere, and ‘V’ is the voltage that is measured in volts.

1. $E_{MC} = T_{MC}.I_{MC}.V_{MC}$ (1.1 - Microcontroller circuit)
2. $E_{ON} = T_{ON}.I_{ON}.V_{ON}$ (1.2 - Microswitch ON state)
3. $E_{OFF} = T_{OFF}.I_{OFF}.V_{OFF}$ (1.3 - Microswitch OFF state)
4. $E_{Switching} = T_{Switching}.I_{Switching}.V_{Switching}$ (1.4 - Switching for Tx and Rx states)
5. $E_{CSMA} = T_{CSMA}.I_{CSMA}.V_{CSMA}$ (1.5 - CSMA circuit)
6. $E_{TX} = P_{Length}.T_{TX}.I_{TX}.V_{TX}$ (1.6 - TX circuit)
7. $E_{RX} = P_{Length}.T_{RX}.I_{RX}.V_{RX}$ (1.7 - RX circuit)

The total energy consumed is then obtained by a sensor node by adding the aforementioned E values as in the following equation. Whereas the first three E values represent packet-independent energy factors (Microcontroller, ON and OFF states); the other four E values represent packet-dependent energy factors (Switching, CSMA, TX and RX).

$$E_{Total} = E_{MC} + E_{ON} + E_{OFF} + E_{Switching} + E_{CSMA} + E_{TX} + E_{RX} \quad (1.8)$$

Consider that there is N number of the sensor nodes in the target area under consideration so the ith sensor with ON state will be responsible for transmission, reception, retransmission, and ACK packets. Using the previously introduced equations can be applied here as follows:

$$E_{TXi} = [P_{TXi} + P_{RTXi}].T_{TXi}.I_{TXi}.V_{TXi} \quad (1.9)$$

$$E_{RXi} = P_{TXi} \times T_{TXi}.I_{TXi}.V_{TXi} \quad (1.10)$$

$$E_{Switchingi} = [P_{TXi} + P_{RTXi} + P_{RXi}].T_{TXi}.I_{TXi}.V_{TXi} \quad (1.11)$$

$$E_{CSMAi} = [P_{TXi} + P_{RTXi} + P_{RXi}].T_{CSMAi}.I_{CSMAi}.V_{CSMAi} \quad (1.12)$$

$$E_{Totali} = E_{MCi} + E_{ONi} + E_{OFFi} + E_{Switchingi} + E_{CSMAi} + E_{TXi} + E_{RXi} \quad (1.13)$$

In summary, each node in WSNs undertakes ‘listen’, ‘transmit’ and ‘receive’ (communication) tasks and switching node’s states ON/OFF. Each of these three actions consumes energy. As every node has a preliminary total energy, once the node starts any of the listening /

communication actions the energy (cost) for undertaking such actions will be deducted from the node's total energy - this manifests the approach presented in this thesis for calculating the nodes' energy which is already published in the literature. In addition, for enhanced simulation-based experimentations and making the results as close to real-life data as possible, the energy space model amplifier (E_{FS}) is included which effectively consumes energy too, as part of the energy consumption calculation.

3.2.3 The RCS Scheduling Algorithm

The rationale of RCS is to group the nodes into K sub-networks, where each subnetwork is scheduled to be ON at a specific time slot. RCS algorithm works as follows:

● In the Initialisation phase: The base station determines the number of sub-networks and randomly assigns each node to a sub-network. The base station then sends out a broadcast message to its neighbour nodes or simply neighbours. This broadcast message contains Node ID, the nodes sub-IDs, Minimum hop count to the base station.

Nodes that are close to the base station receive this Broadcast message and allow a backoff delay timer. For example, after a node receives the broadcast message, it allows some time delay before it updates its hop count in case another message is received from a closer neighbour. Without the backoff delay, a closer neighbour might be ignored.

● Initial hop count propagation: The hop count is propagated from the base station. The base station hop count is 0. Now, the nodes that receive the broadcast message from the base station increment their hop count by 1 and rebroadcast the message to their neighbours. The latter do the same process when they receive the broadcasted message, and so forth through the whole network.

● Determining the sleep schedule: After a node exchanges information with its neighbours, each node knows its sub-network and the specific time to be ON and OFF.

● Determining the upstream nodes: Each node does a simple processing in order to select its upstream neighbour node. The node that sends the broadcast message first would be the one selected as the default upstream node. This decision is maintained during the initialisation phase. For example, assuming there are 4 nodes A, B, C and D and 1 base station, where nodes B and C are the upstream nodes of node D. The 4 nodes are scheduled to be ON respectively at time slot 1, 2, 3 and 4. When nodes exchange information, there will be two cases:

● Nodes B and C are neighbours: if B broadcasts its message to C first, then C would know that B is also an upstream node of D, then it would be ON during its original time slot only, i.e., time slot 3. However, node B would be scheduled to be turned ON in time slots 2 and 4.

● Nodes B and C are not neighbours: In this case, both nodes will be ON in time slot 4 in addition to their time slots, i.e., node B will work in time slot 2 and 4 and node C will work in time slot 3 and 5 as illustrated by Figure 3.2.

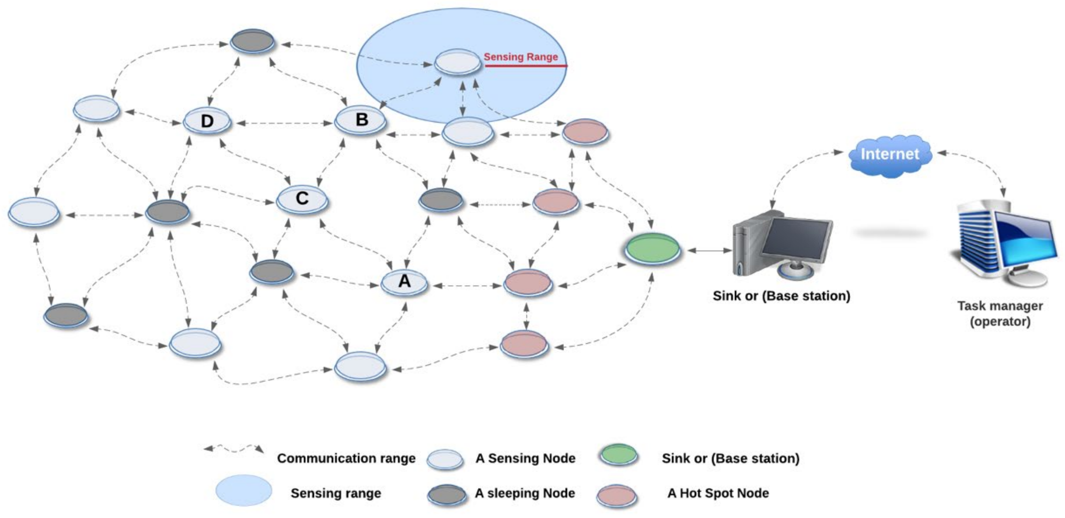


Figure 3.2 An example of how nodes are tuned ON and OFF by RCS.

● Set the schedule, the sub-networks will alternate between ON and OFF periodically.

Figure 3.3 represent the main functionalities of RCS algorithm.

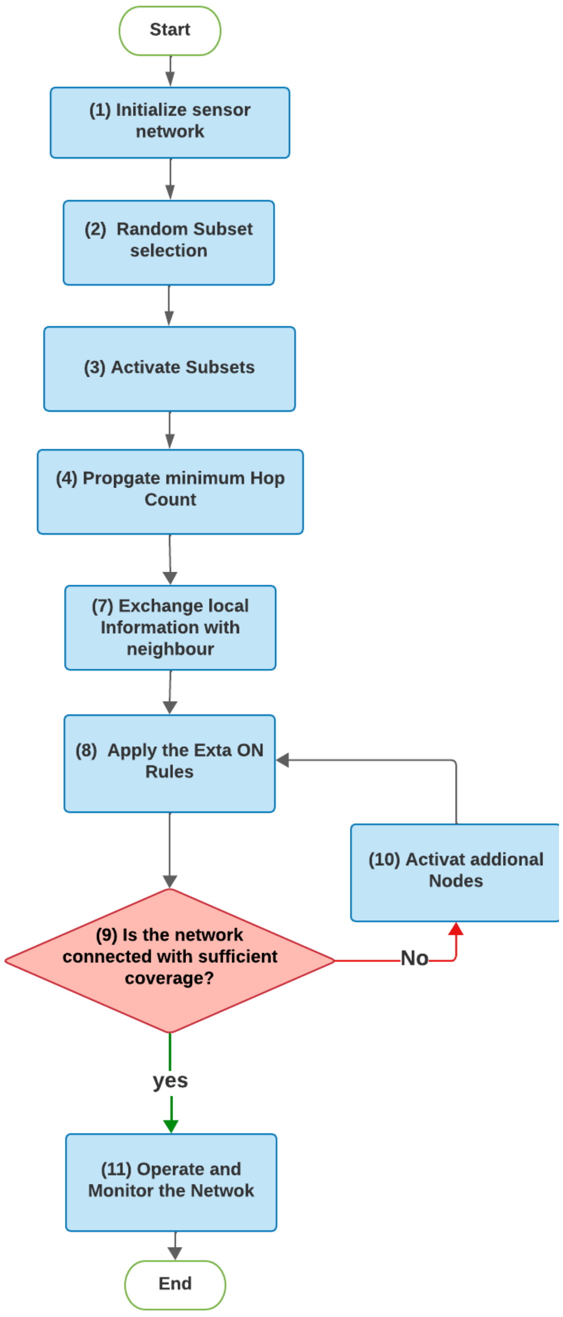

Start
(1) Initialize sensor network
(2) Random Subset selection
(3) Activate Subsets
(4) Propgate minimum Hop Count
(7) Exchange local Information with neighbour
(8) Apply the Exta ON Rules
(10) Activat addional Nodes
(9) Is the network connected with sufficient coverage?
No
yes
(11) Operate and Monitor the Netwok
End

Figure 3.3 The main functionalities of RCS algorithm.

The following is the pseudocode of Algorithm 3.1 that summarises the main functionalities of RCS algorithm:

Algorithm 3.1 The RCS Algorithm

Input: Set of sample data of Sensor Nodes

Output: Sensor nodes operates with sleep awake schedule

```
1: Start
2: Initialize Sensor Network:
3: Deploy sensor nodes randomly.
4: Divide sensor nodes into multiple disjoint subsets.
5: Random Subset Selection:
6: Each sensor node randomly selects and joins one of the subsets.
7: Activate Subset:
8: Activate one subset while keeping others inactive to conserve energy.
9: Propagate Minimum Hop Count:
10:  Each node calculates its minimum hop count to the sink node.
11:  Propagate the hop count information to neighbouring nodes.
12:  Exchange Information with Neighbours:
13:  Nodes exchange their hop count, subset membership, and
     upstream/downstream information with their neighbours.
14: Apply Extra-on Rule:
15: For each node: - If the node's downstream neighbour is active and no
    upstream neighbour is active:
 6: Activate the current node outside its scheduled time slot.
17:  Check Connectivity and Coverage:
18: If the network is connected and coverage is sufficient: - Proceed to the next
    step
19:  Else: - Activate additional nodes and repeat the extra-on rule.
20: Operate and Monitor:
21: Nodes operate according to the finalized schedule.
22:  Monitor network performance for connectivity and coverage.
23: End
```

3.3 Experimentations Set-Up

Two sets of experimentations were performed. The motivation behind such experiments is to observe the behaviour of the WSN with and without any supporting scheduling algorithm i.e., having the nodes "turned ON" throughout the experiment time. This will allow us to evaluate how nodes consume energy and behave in the deployed environment. This is to manifest how crucial scheduling behaviour for dependable WSNs has such an impact on network lifetime, and hence service availability and reliability. This can be used to highlight what is analysed in the literature regarding the importance of preserving network lifetime via manipulating nodes' states ON/OFF i.e., scheduling. During the test-based portion of the experiment, a neural network will be supplemented into the system.

3.3.1 Experiment 1: WSN without A Scheduling Algorithm

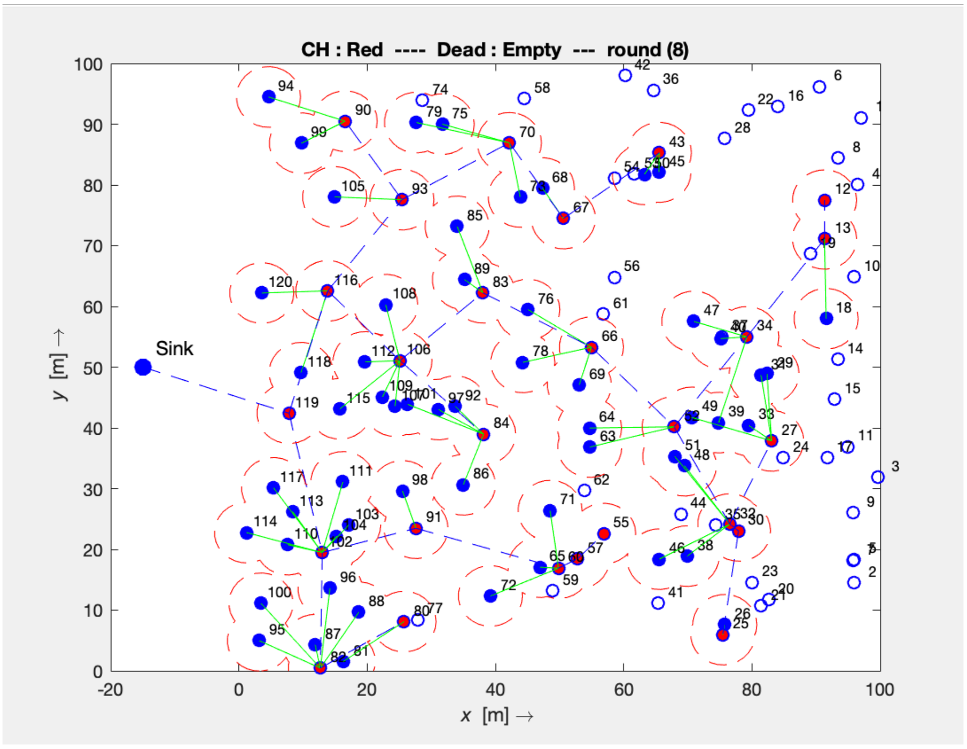


Figure 3.4 Simulation Set-up of WSN without Scheduling Algorithm.

In the simulation, 1 stationary Sink Node and 150 stationary homogeneous sensor nodes are considered distributed randomly in a 100 by 100 m2 field as the area to be monitored as shown in Figure 3.4. The experimentation is based on executing multiple simulation rounds. The number of rounds is set to 2000 as the upper bound. The simulation will halt execution once

95% of the nodes reach complete energy depletion i.e., nodes break point. The sensing range of each sensor node is set to 5 m. The maximum communication range per sensor is set to 15 m. The network lifetime is calculated by the consumption of communication (transmitting and receiving), and sensing units of each node. Refer to Table 3.1 for a tabulated form of the aforementioned parameters.

In the beginning of the simulation, each node starts its operation with an initial energy of 3 joules. Every sensor node consumes 1.0000e^{-09} joules of its initial energy when transmitting or receiving one bit. In addition, amplifying the signals i.e., digitising them for transmitting to/from nodes in free space consumes 5.00e-08 joules energy. As the simulation progresses in time, there will be more energy consumption due to nodes activities i.e., sending, receiving, sleeping, and awaking etc, which will have effects on network lifetime, connectivity, and coverage.

To the author’s best knowledge, factoring in this specific energy consumption element has not been considered in any work by previous authors. This adds more novelty and contribution to this chapter in the WSN community.

3.3.2 Experiment 2: WSN with RCS Scheduling Algorithm

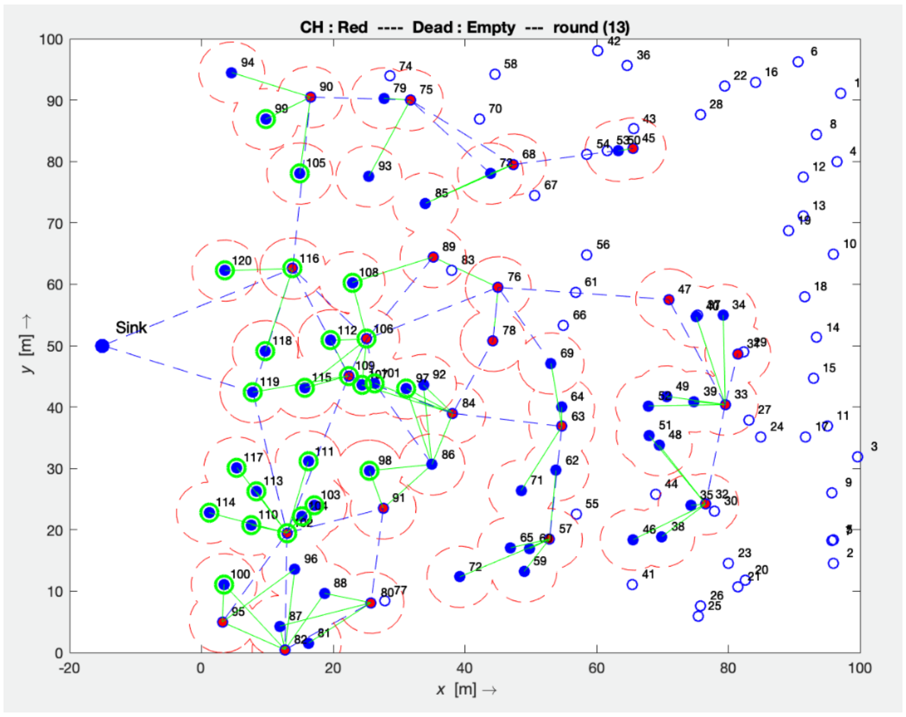


Figure 3.5 Simulation Set-up of WSN with RCS Scheduling Algorithm.

The motivation behind this experiment is to evaluate the behaviour of nodes when a scheduling algorithm is implemented to turn nodes ON and OFF with respect to optimisation objective(s). In this experiment, albeit using the same configurations of Experiment 1 above, a new parameter called K which represents the number of subsets of nodes in the network was added to the simulation. This K parameter is introduced by the RCS algorithm where K divides the nodes into a number of K subsets. Each subset alternates between On and Off states to preserve energy and extend the network lifetime. In this experiment, the parameter K is set to 3. Figure 3.4 above presents the simulation setting.

3.4 Experiments Results

This section represents the key metrics in which the simulation results are obtained, the focus will be on all used energy, lifetime of sensor nodes, efficiency, connectivity, and coverage.

3.4.1 All Used Energy

The energy consumption of each node is dramatically higher without the scheduling algorithm as in Experiment 1, even managing to cease activity at 640th rounds. In contrast, when the RCS scheduling algorithm is used in Experiment 2, each node uses substantially less energy and remains active for 1180th rounds. This shows that the WSN application uses less energy (energy efficient) in the absence of an RCS scheduling algorithm. This is due the exploitation of node redundancy in utilizing the scheduling mechanism where nodes are alternating between ON and OFF states, thereby, the optimising the sensor nodes energy level for better performance.

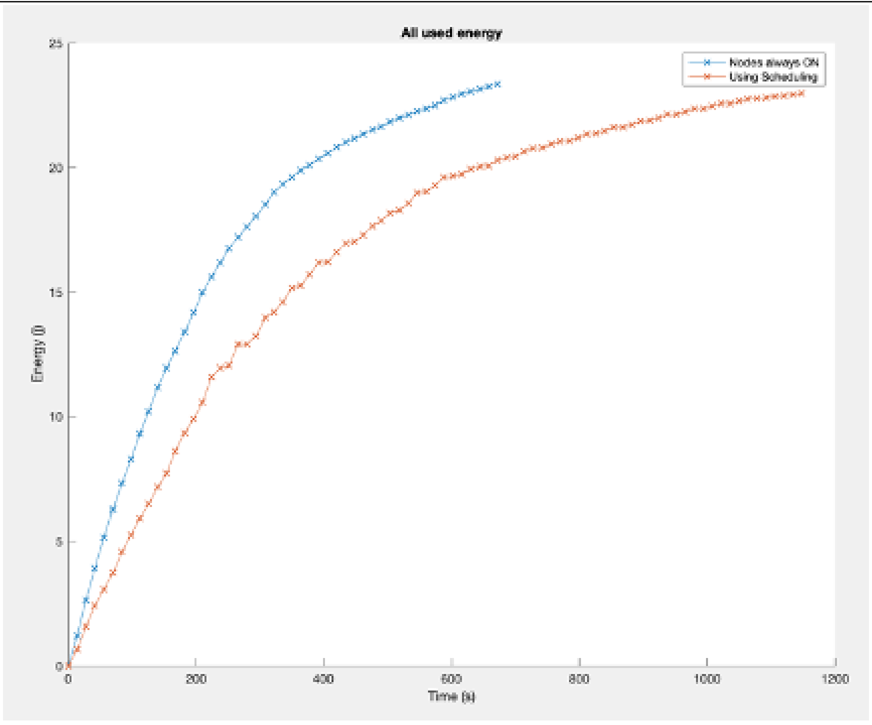


Figure 3.6 All Used Energy Lifetime of a Sensor Node

Figure 3.6 presents the number of nodes alive at any given round. The number of active nodes starts dropping at 5th rounds, 50% of nodes have ceased to be alive at rounds 23, and 100% are dead at 49 rounds. In contrast, in Experiment 2, 50% of nodes have died at round 34, and 100% at round 85.

3.4.2 Lifetime Sensor Nodes

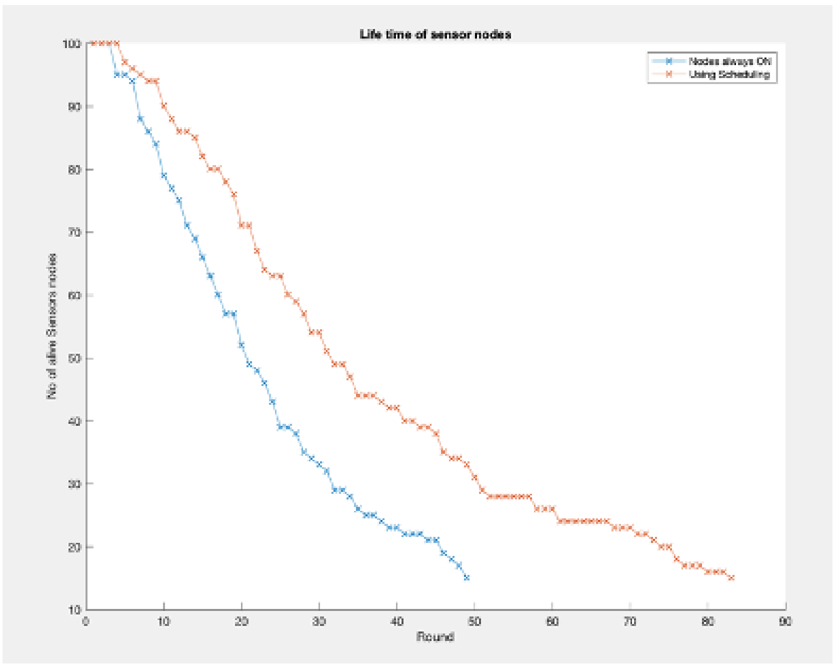


Figure 3.7 Lifetime of Sensor Nodes.

As can be noted from the blue curve Figure 3.7, the number of active nodes starts dropping at round 5, where 50% of nodes cease to be alive at rounds 23, and 100% are dead at round 49. In contrast, in Experiment 2, an orange curve where nodes are scheduled to be ON and OFF, thereby participating in network activities but preserving energy when they are in an OFF mode. As can be noted from the orange curve, the number of active nodes starts dropping at round 7, where 50% of nodes died at round 34, and 100% at round 85. Hence, Experiment 2 when node scheduling is used the number of a live sensor remains for a longer period of time (rounds), in composition to Experiment 1 which clearly shows an improvement in network lifetime.

3.4.3 Efficiency

The efficiency of the data packet passed through 100 sensor nodes over 1200 seconds. Figure 3.8 shows that the efficiency is lower for nodes activated in the absence of a scheduling algorithm because the bandwidth consumption is higher for non-scheduled nodes. As for the scheduled nodes there is appropriate usage of bandwidth, hence reflects the efficiency of the scheduling nodes. This is shown in the Figure 3.8 as for non-scheduled algorithm the data packet generation stopped at the rounds 600th and for scheduled algorithm at 1100th rounds.

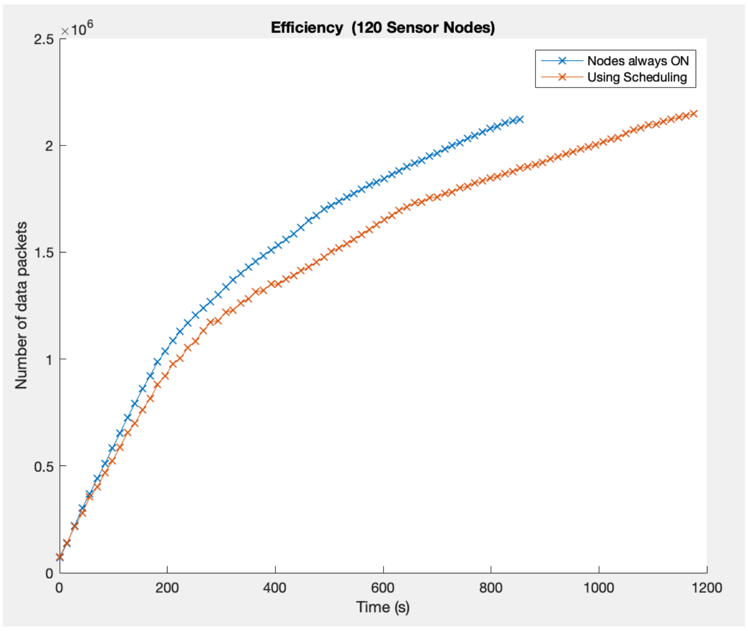


Figure 3.8 Efficiency

3.4.4 Connectivity

During 90th rounds, the nodes' connectivity was tested separately using a scheduling algorithm (experiment 2), and without using one (experiment 1). The results show that connectivity efficiency is dramatically reduced when the system is not supplemented with a scheduling algorithm, as the nodes cease to run after 50th rounds. The data provided in Figure 3.9 show service availability is maintained for a longer duration when a scheduling algorithm is present in a system. This attributed due the fact that non-scheduled nodes are not well managed, as a result they tend to expend energy in burst mode and therefore using the max part in setting up the network and very less of useful energy to run the connectivity functions.

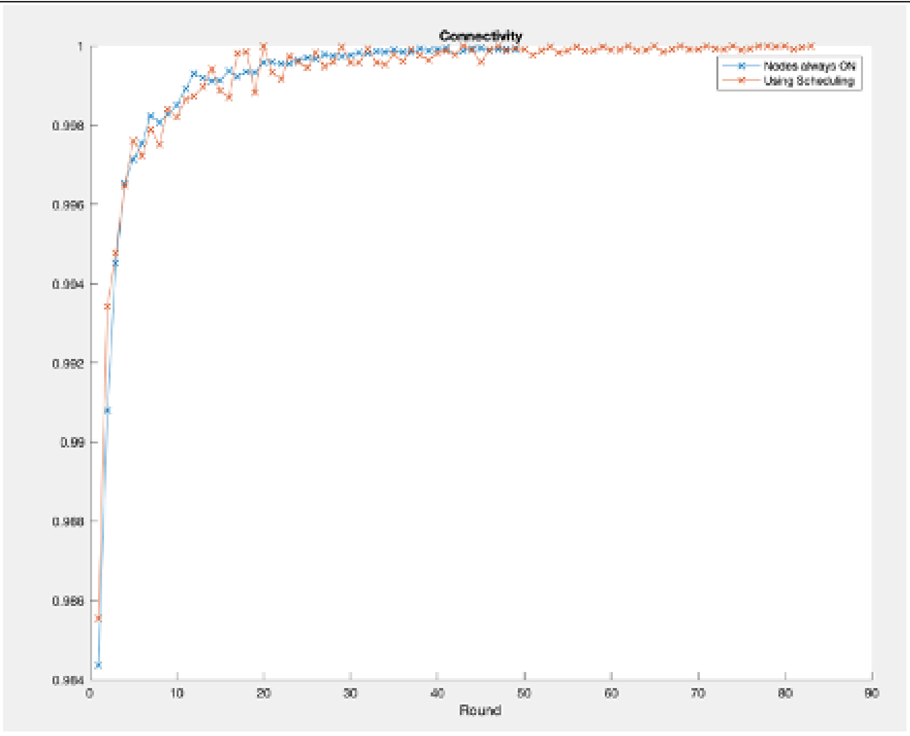


Figure 3.9 Connectivity

3.4.5 Coverage

Figure 3.10 depicts the ratio of coverability for a sensor node to monitor, using a scheduling algorithm (experiment 2) and in its absence (experiment 1). In conjunction with Figure 3.9, once the nodes cease to be connected, their coverage will drop without a scheduling algorithm. This is possible because the RCS algorithm identifies some sensor nodes from the population of 100 nodes to be more active than others so that we could improve the performance and hence obtain better coverability in the network. In Figure 3.10, the rapid reduction in the coverage from 0 to 10th round is due to the initial setup process and after the setup process is stabilized the coverage functions are back in full performance as per the condition's setup in the scheduling and non-scheduling algorithm.

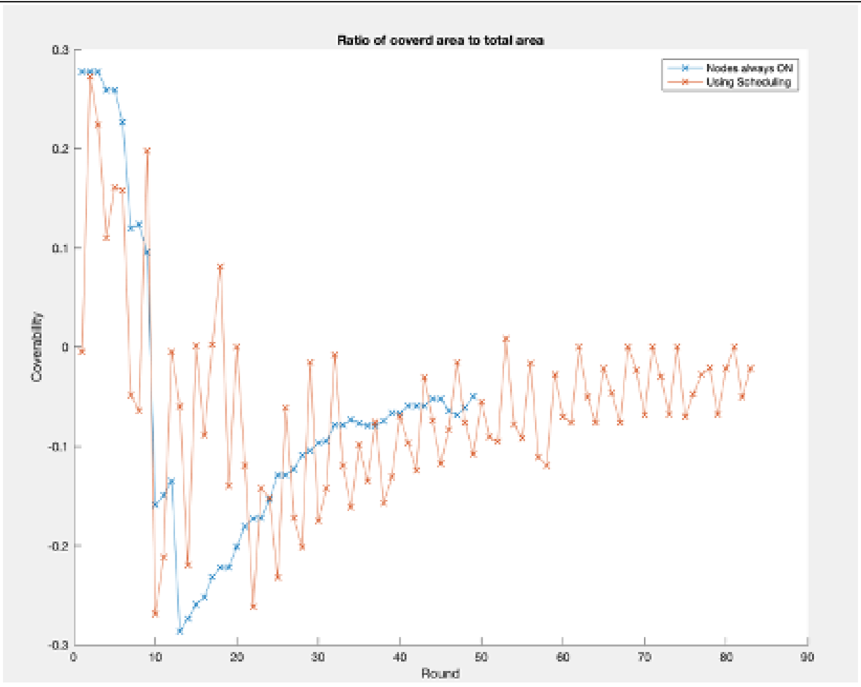


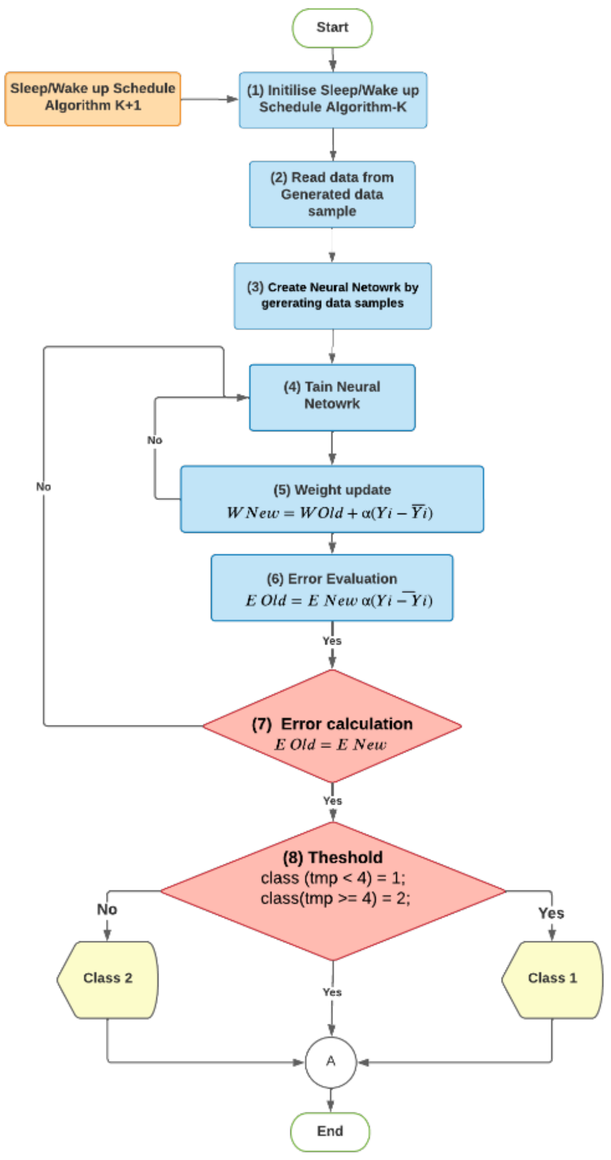
Figure 3.10 Coverability

It is clearly noted from the Figure 3.10 above that the none scheduling algorithm coverage stopped at 50th rounds this is because nodes are always active in an ON state. Hence, the level of energy consumption is higher in contracts with the use of scheduling algorithm where nodes are not always ON, but alternating between ON and OFF states where the level of energy consumption is low the coverage is maintained until the 85th rounds. Hence, more time in performance and improved level of services availability and reliability in the WSN.

3.5 A Perceptron Multi-Layer Neural Network

This section presents the proposed solution and the simulation of the RCS algorithm [21] in MATLAB. The elements of the simulation are the RCS scheduling algorithm, the Perceptron ANN-based analysis tool, and a generic WSN application. Figure 3.11 below illustrates the basic framework of the analysis of the Scheduling Algorithm:

1. Initialise the Sleep/Wake up Schedule Algorithm via MATLAB, with randomised Node deployment.



Start
Sleep/Wake up Schedule Algorithm K+1
(1) Initilise Sleep/Wake up Schedule Algorithm-K
(2) Read data from Generated data sample
(3) Create Neural Netowrk by gererating data samples
(4) Tain Neural Netowrk
No
No
(5) Weight update
$W New = W Old + \alpha(Yi - \overline{Y}i)$
(6) Error Evaluation
$E\ Old = E\ New\ \alpha(Yi - \overline{Y}i)$
Yes
(7) Error calculation
$E\ Old = E\ New$
Yes
(8) Theshold
class (tmp < 4) = 1;
class(tmp >= 4) = 2;
No
Yes
Class 2
Class 1
Yes
A
End

Figure 3.11 Proposed Framework

2. Apply a Neural Network Schedule Analyser to predict which Nodes are activated more often, given the number K of groups.

3. Evaluate the data collected to ensure that the error rate (accuracy of classification) is withheld to a decent percentile: if the error is not within an acceptable range, another scenario is commenced to increase the sample size of data, as continuous testing allows the neural network analyser to predict with a higher level of accuracy.

The proposed ANN analyser is a computational solution used to process, analyse, and interpret data using Artificial Neural Networks model designed specifically for node scheduling in WSNs. The ANN analyser is composed of the following functions:

- Normalization: Scaling input data to a standard range, often between 0 and 1. For example, it can normalize the data produced by WSN.
- Encoding: Converting categorical data into numerical formats.
- Feature Extraction: Identifying and extracting relevant features from raw data to improve model performance.
- Prediction: Using the trained model to predict new, unseen data and providing valuable insights and predictions that are crucial for decision-making in various domains.

It is worth mentioning that there are several commercial and open sources ANN tools, but they are built for very general purpose and not specific for our problem domain. In addition, it is very difficult to reverse engineer these general tools to build functions that suit our specification requirements.

3.6 Problem Formulation

Perceptron supervised learning algorithm is a simple version of artificial neural networks. The basic process is to train the given input X with the output Y mapping and then test it with some data samples. It is a supervised algorithm as it supervises the outputs mapping data Y. The main goal is to convert the data into two classes: class 1 and class 2. As shown in Figure 3.12.

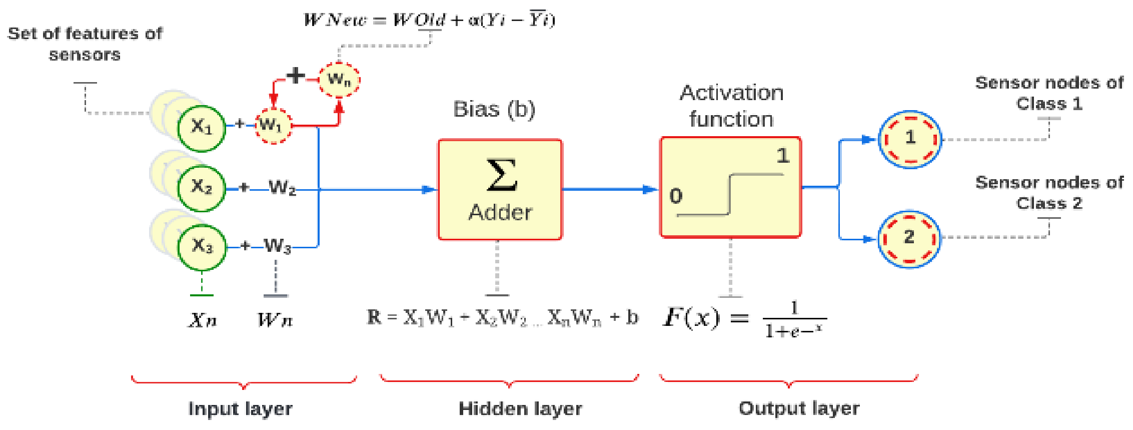


Figure 3.12 Perceptron-based Neural-Network Algorithm

Hence, a perceptron algorithm is known as a binary classifier as shown in equations (3.4) and (3.5). The following equation represents the data structure of the perceptron algorithm in the WSN scheduling algorithm, the general structure of the perceptron algorithm is borrowed from [Kubat, 1999].

$$R = i1nXiWi + b \qquad (3.1)$$

Where R is the results, sigma is the summation of all intended X features, and the associated weight W. n represents the number of features and b is the bias value. In the main problem of concern domain X represents the position of each sensor node in the network to the Base Station. Simplifying the above equation will give us the following result:

$$R = \sum X . W + b \qquad (3.2)$$

The following function called the activation function, takes the input of equation (3.2), R, and produces the binary classification.

$$\boldsymbol{F(x)} = \frac{1}{1+e^{-x}} \qquad (3.3)$$

The output of the activation function is of either 1 or 0. In the assumed scenario it is called class 1 or class 2.

$$(X . W + b) > 4 = class\ 1 \qquad (3.4)$$

$$(X . W + b) < 4 = class\ 2 \qquad (3.5)$$

The above process of training goes through multiple iterations to reach zero error of the smallest possible error value.

$$W\text{New} = W\text{Old} + \propto (Yi - \overline{Yi}) \qquad (3.6)$$

The update function adds the new weight (Wnew) to, the learning rate, multiplied by the difference of the right Yi and Yi wrong output which is the error rate. The weight is updated until the desired solution is achieved. Note, the value of the error rate which influences the learning time.

The following is the pseudocode of Algorithm 3.2 that summarises the proposed solution:

Algorithm 3.2 Perceptron-based Neural-Network Algorithm

Input: Set of sample data of Sensor Nodes

Output: Set of sample Sensor Nodes of Class **1** or Class **2**

```
1: Initialize:  Sleep Awake Cycle (k to k+1)
2: Read data from the data sample as ~x
3: Train Neural Network ()
4: Choose an initial weight vector ~W
5: Initialize the minimization approach
6:  While the error did not converge do
7:  For all (~x, ~d) € D do
8:      apply ~x to the network and calculate the network output
9:      calculate error (~x)
10:  end For
11:   calculate error (D)
12:  For all weights summing over all training patterns
13:   perform one update step of the minimization approach
14:  end while
15: if Threshold > 4 then
16:     Class 2
17:  Else
18:     Class 1
19: end if
```

3.6.1 Experiment 3: WSN with RCS Scheduling Algorithm Supported by ANN

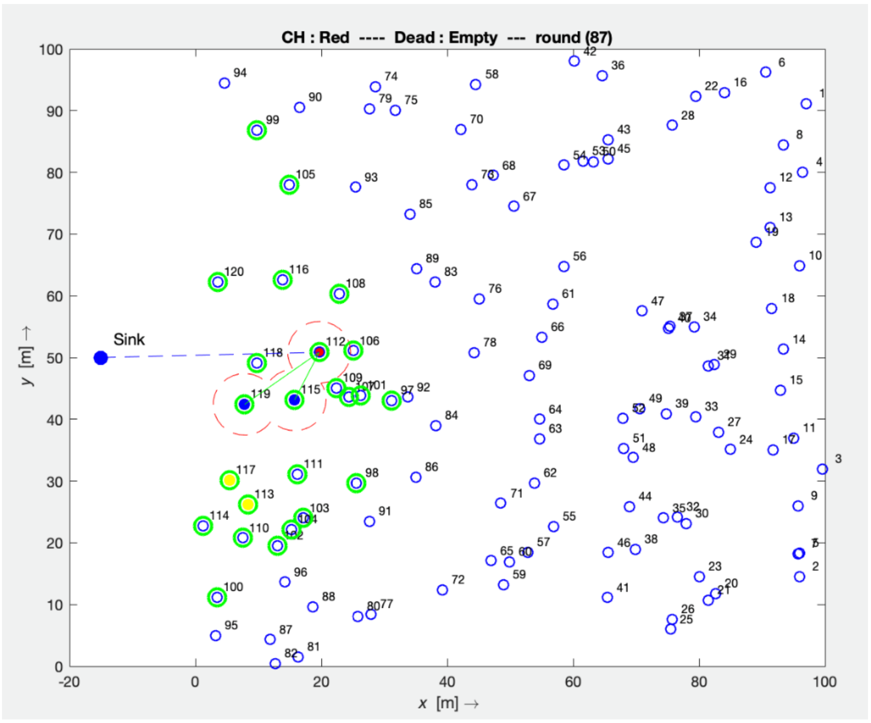


Figure 3.13 Scheduling Algorithm for WSNs with the use of Neural Networks

The purpose of the ANN is to analyse the performance of the scheduling algorithm by returning the number of nodes that are overworked. The ANN returns for each node if it is overworked (class 2) or not (class 1). The data collected from the ANN is used to further analyse and enhance the dependability of the WSN. Figure 3.13 above present the simulation environment in which, the ANN analyser applied to analyse the RCS scheduling algorithm. As can be noted, the overwork node is highlighted in green. Table 3.2 represents the new ANN parameters that were generated from the simulator.

Table 3.2 Parameter of the Perceptron Artificial Neural Network Analyser

Neural Network Parameters	Value
NumToCreate (# of data sampled)	1000-2000
R (Maximum Range of Transmission)	15
S X (X-Axis Coordinates of Sink Node)	-15
S Y (Y-Axis Coordinates of Sink Node)	50
Number of Input Variables	2
Number of Functions to fit	1
Ratio of Data used for creating model	0.85
Error Rate @ 200^{th} Iteration	4.990006e-01%
Accuracy of Classification @ 200^{th} Iteration	86.1302

3.6.1.1 Training Progression

Figure 3.14 depicts the neural network's training progression, and the error rate changes across every consecutive iteration of the training, ending at 200 iterations. It is worth noting that the training depends upon the amount of data taken into consideration with the fine tuning of hyperparameters. During the simulation process, the error rate reaches an acceptable range during the 149th iteration, holding a value of 4.998903e-01 (0.005%), where the final stabilised error rate at the 200th iteration is 4.990006e-01% (0.005%). Those value were generated from the simulator.

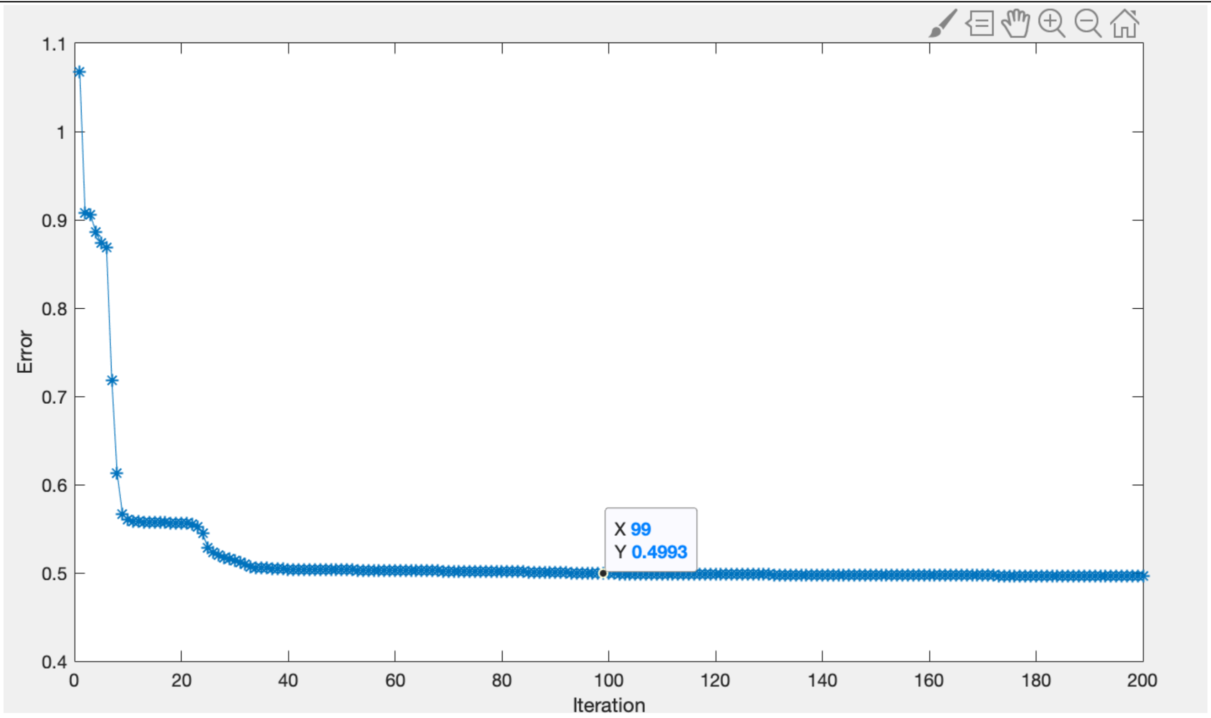


Figure 3.14 Training Progression

The accuracy of classification for the training is 86.130, meaning the neural network could detect which nodes will 'Extra On' in four or more occurrences throughout the 200 iterations of testing taken place in the training segment of the experiment.

3.6.1.2 Neural Network Analysis

Neural Network Analysis: The experiment was undertaken to emphasise the scheduling algorithm's strengths; a simulation was performed using a pre-existing replicated RCS within MATLAB, which was built from the ground up as there is no open-source framework for this algorithm and one simulation in the absence of the RCS. The algorithm in question was chosen due to its ability to consider coverage, connectivity, and a lifetime of a network regarding wireless sensor nodes. Coverage, connectivity, and a lifetime of a network were used to measure the wireless sensor network's performance, as shown in the figures presented above as matrices. It was clear that the RCS's presence substantially increased a system's performance in the parameters mentioned earlier throughout every matrix.

Evaluation of the ANN analysis: Following our experiment, the ANN algorithm returns 19 Nodes that died earlier than any other node in the network (Overused). Figure 3.13 shows all 19 Overused Nodes (Highlighted in green colour), which are situated in the 1-Hop area. This will create a partition in the network whereby events will be detected in the subsequent k-Hop areas (k > 1) but will not be reported to the Base station, thus rendering the whole implementation of the WSN useless and undependable. The ANN analysis hence confirms the initial hypothesis that nodes that are closer to the Base station are more likely to undertake more workload. However, the ANN analyser is limited to discovering the number of Overused nodes and where partitioning is more likely to occur in the network but does not provide a way to improve the existing algorithm in question (e.g., RCS in this case) or come up with a new suggestion to revive the 1-Hop area, e.g., by utilising nodes in the 2-Hop area instead.

3.7 Evaluation and Interpretation

In this chapter, an ANN Analyser has been developed to analyse the scheduling performance and suggest possible solutions with respect to state space exploration for the MOO problem. The simulation results have shown limitations in the mechanism of the algorithm itself in this case the RCS-based scheduling algorithm. Hence, indicates there is more room for state space exploration to obtain a better solution. For example, the randomisation of the RCS algorithm mechanism partitions the network into two or three segments. In particular, a partition can occur close as well as further away from the base station, as the death of subsequent nodes can result in active nodes attempting to compensate for the drop-in coverage and connectivity in the system.

Hypothetically, there are possible ways to improve the lifetime of these sensor nodes by using a more dynamic connectivity/coverage strategy, which can involve increasing the transmission radius of certain nodes to balance the load on 1-Hop Nodes. Another possible solution would be further to divide the area into 1-Hop, 2-Hop N-Hop areas, and determine the best trade-off between all these areas (in terms of connectivity and coverage) that helps improve Network lifetime.

It is worth mentioning, that the ANN analyser doesn’t provide the best solution, but it tells the limitations of the algorithm under testing in this case the RCS. In specific, it confirms if the solution is optimal and why if it is not. Now that the methods to analyse scheduling algorithms using ANN analyser to train then test the performance of the scheduling algorithm [Liu et al., 2006] is obtained. The research methodology is to investigate different techniques for state space exploration to find the better or optimal solution for the MOO problem in WSNs. In addition, the emphasis is to find an approach that provides the solution or a method to confirm that the obtained solutions are the right or optimal solutions.

The methodology stems from ANN Analysis outcome in which more state space exploration is needed to find possible avenue that would lead to the near optimal or optimal solution for the MOO problem. Thereby, improves the WSN system’s overall dependability.
There is a promising avenue in term of Bio-inspired computation, and Machine learning approach. There are many methods and approaches to address state space exploration. For example, Bio-inspired based Bat approach and Machine learning using spatial and temporal approach is selected to address the state space exploration. However, from the outset of it cannot be determined which approach would provide the best possible solution. Therefore, to ensure which algorithm should give the best possible solution.

Chapter 4 Using Hidden Markov Chain for Improving the Dependability of Safety-Critical WSNs

4.1 Introduction

The main aim of the ANN analyser is to explore the state space and find a unique solution to the given problem regarding connectivity, coverage, and energy efficient in node scheduling mechanism. For example, in chapter three the ANN analysis confirms the limitation in the RCS algorithm mechanism itself [Liu et al., 2006]. For instance, many unique solutions can be explored to provide a better solution which shall improve the WSN's lifetime, and service continuity, reliability, and availability. Specifically, through the ANN analysis, scenarios were captured, and each round discovered that some nodes are over-worked, while others under-worked, hence the network's service availability and lifetime are compromised. The reason for this is because nodes are not working in an efficient manner, this level of non-determinism makes optimal load balancing for future performance optimization a challenging task due to the randomized nature of the RCS algorithm. For example, towards the end of the network life cycle, some nodes are found to be already obsolete (dead), this can lead to breaking the WSN into different segregations due to nodes' absence/death. The result of this situation can lead to data loss due to inability to transmit messages across the WSN because of the dead nodes.

ANN analysis [Chapter 3] has shown for a simple WSN example how in each round of the simulation, some nodes are overworked while others are underworked, eventually leading to an early partition of the network whereby 1-hop nodes die first and no communication can reach the base station. Consequently, many unique solutions/schedules can be explored that improve on the performance of RCS initial (random) schedule. However, these unique solutions are determined in an ad hoc way such that a further step is yet needed to find amongst them which one (or set thereof) provides the best solution.

The purpose of utilizing HMM to lengthen the WSN service availability by delaying reaching the state of nodes' obsoleteness via optimal load balancing. Reaching a state of obsoleteness is an inevitable state for nodes, and WSNs by and large. However, delaying reaching this state has service, and arguably economic, benefits. HMM was utilised to intelligently extend the usability of nodes to collectively increase the service availability and reliability of the WSN. The author of this thesis acknowledges to the best of his ability that the literature lacks focusing on extending the WSN lifetime in the best, useful, and most possible meaningful way. Hence, exploring the state space to find such a near-optimal solution is important. Related work in literature has focussed on extending the last remaining node alive, while having aimed for extending the collective lifetime of the WSN.

In this chapter, a HMM is proposed to model the behaviour of a WSN. An optimal solution can then be achieved by ensuring that each node is used for transmission dependent on its current energy level. As the latter decreases, the probability for the node to be in a transmit state decreases while its probability of being in a receive state increases. Notably, a new algorithm is introduced that ensures the assigned probabilities are the best possible. Unlike the RCS algorithm, the HMM-based algorithm returns the energy levels of every node and triggers maintenance warning messages such that dying nodes can be replaced in due time.

4.2 HMM and its Usability in WSNs

The HMM Model is used to optimize the ON/OFF states by analysing the OFF states and the ON states as well as the IDLE states. The latter also consumes energy via memory buffering and CPU processing of already stored data. More specifically, in this chapter, the number of Transmit (Tx) and Receive (Rx) states were balanced in the nodes ON states. As known, Tx states are more expensive, energy consumption wise, than Rx states.

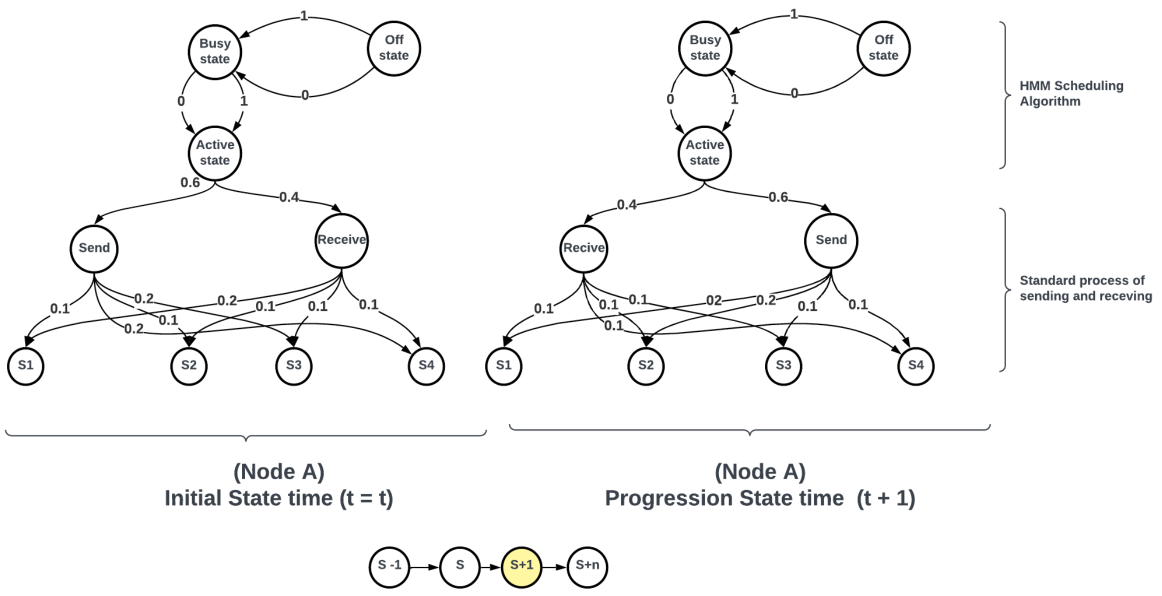


Figure 4.1 Balancing Tx and Rx states using HMM.

Figure 4.1 illustrates a simple example of one sensor node (A) that has two main functions (transmit and receive) where the probability for both states is equal (0.5,0.5). As there are hidden states like s1, s2, s3, and s4 in the above diagram, their initial probability values would be (⅛ + ⅛) per state. It is noticeable that all states have equal probability values, but in the long run, and as the energy depletes, the receive-state will take less energy so it will have higher probability and, hence, the more expensive it will be for transmit-state to take place. This will result in assigning less probability values to transmit-states, then more states will behave as a receiver rather than a transmitter.

Consider the WSN application from Chapter 3. Figure 4.1 illustrates the HMM of a node from that WSN, showing on the left the initial probabilities assignment and on the right the probabilities after 800 rounds. Between 800-850 rounds, 10% of nodes function as transmitters only; and between 801-820 rounds, 80% of nodes may act as a transmitter node but they rest afterwards until the 850th round by acting as receiver nodes.

4.3 The HMM Algorithm

The transmit and received states were split into four probability states which divides the transmission states as well as the receiving states into four transition probability states as shown in the Figure 4.1 above. The HMM optimization uses these transmission probabilities for switching into four different states for granularization (e.g., stepwise refinement splitting the big problem into small pieces) using the HMM functions. The HMM function assigns predefined probabilities as they are above (⅛, etc.) per transition probability.

A sensor node initially has two states, but now after using HMM a senor node will be assigned four states (S1, S2, S3, S4) as illustrated in Section 4.2; the greatest probability amongst S1, S2, S3, S4 will be assigned the sensor node instantaneously. This means, as the network progresses in time, energy of each node reduces, hence probability using HMM is a good fit to keep up with which nodes to transmit and which nodes to receive. As time evolves, the probability of receiving is higher than transmitting. For a WSN of five nodes: A, B, C, D, and E. Not all nodes will be classified as Tx and Rx nodes, but only those whose energy decreases in time at a rate that the WSN's base station decides to transition them to be Rx nodes only. However, there will remain Tx nodes too, depending on their current predicted energy levels.

Algorithms 4.1 represents the Pseudo code functionality of the HMM node scheduling algorithm.:

Algorithm 4.1 Pseudo code for the HMM Algorithm

Input Parameters:	X: the input domain of the function to optimize (Sleep, awake, transmit, receive)
	F: the function to optimize (optimised s-w-t-r cycles)
	N: the number of hidden states in the HMM
	T: the length of the observation sequence (suppose there is (S, W, T, R, ...))
	max_iterations: the maximum number of iterations for the optimization algorithm
Output Parameters (epsilon):	the threshold for convergence (minimum number elements in the sequence e.g., S, R)

```
1. Initialize the HMM:
   - Randomly initialize the initial state probabilities, transition probabilities, and emission probabilities.
2. Initialize the observation sequence:
   - Generate a random sequence of length T by selecting inputs from the input domain X.
3. Run the Viterbi algorithm:
   - Use the Viterbi algorithm to find the most likely sequence of hidden states given the observation sequence.
4. Update the model parameters:
   - Update the initial state probabilities, transition probabilities, and emission probabilities based on the most likely sequence of hidden states found in step 3.

5. Repeat steps 3 and 4 until convergence or until the maximum number of iterations is reached:
```

- **Generate** a new observation sequence using the updated model parameters.
- **Run** the Viterbi algorithm to find the most likely sequence of hidden states given the new observation sequence.
- **Update** the model parameters based on the most likely sequence of hidden states found in step 5.
- **Check** for convergence by comparing the log-likelihood of the observation sequence between iterations.

6. **Return** the optimized function:
   - The optimized function is given by the emission probabilities of the final HMM.

Viterbi Algorithm:

```
function Viterbi (Observations, States, Transition_Probabilities, Emission_Probabilities):
    T = length of Observations
    N = length of States
    V = matrix with dimensions (N x T) initialized to 0
    Backpointer = matrix with dimensions (N x T) initialized to 0
    # Initialization
    for i in range(N):
        V[i][0] = Transition_Probabilities[0][i] * Emission_Probabilities[i] [Observations [0]]
        Backpointer[i][0] = 0

    # Recursion
    for t in range (1, T):
        for i in range(N):
            max_prob = -1
            max_state = -1
            for j in range(N):
                prob = V[j][t-1] * Transition_Probabilities[j][i] * Emission_Probabilities[i][Observations[t]]
                if prob > max_prob:
                    max_prob = prob
                    max_state = j
            V[i][t] = max_prob
            Backpointer[i][t] = max_state
    # Termination
    max_prob = -1
    max_state = -1
    for i in range(N):
        if V[i][T-1] > max_prob:
            max_prob = V[i][T-1]
            max_state = i
```

```
# Backtracking
Path = [max_state]
for t in range (T-1, 0, -1):
    max_state = Backpointer[max_state] [t]
    Path.append(max_state)
Path.reverse()
return Path
```

4.4 Experiment Results

As described in Chapter 3, the same experimental set-up is implemented. The main objective of this experiment is to analyse the performance of the new proposed HMM algorithm with RCS scheduling algorithm. The proposed solution can be applied to any scheduling solution since it alerts the state of the ON node at a very particular point in time at a very particular event, e.g., when a node's level of energy reaches a certain level. In this experiment, the new HMM Node scheduling algorithm was analysed using the MATLAB simulation environment. The experiments were set up in a 100m^2 area to be monitored. The simulation consists of 100 stationary sensor nodes that are randomly deployed. The base station is located on the far-left edge of the network. All nodes are homogeneous which means that they have the same sensing and communication capabilities as depicted in Figure 4.2.

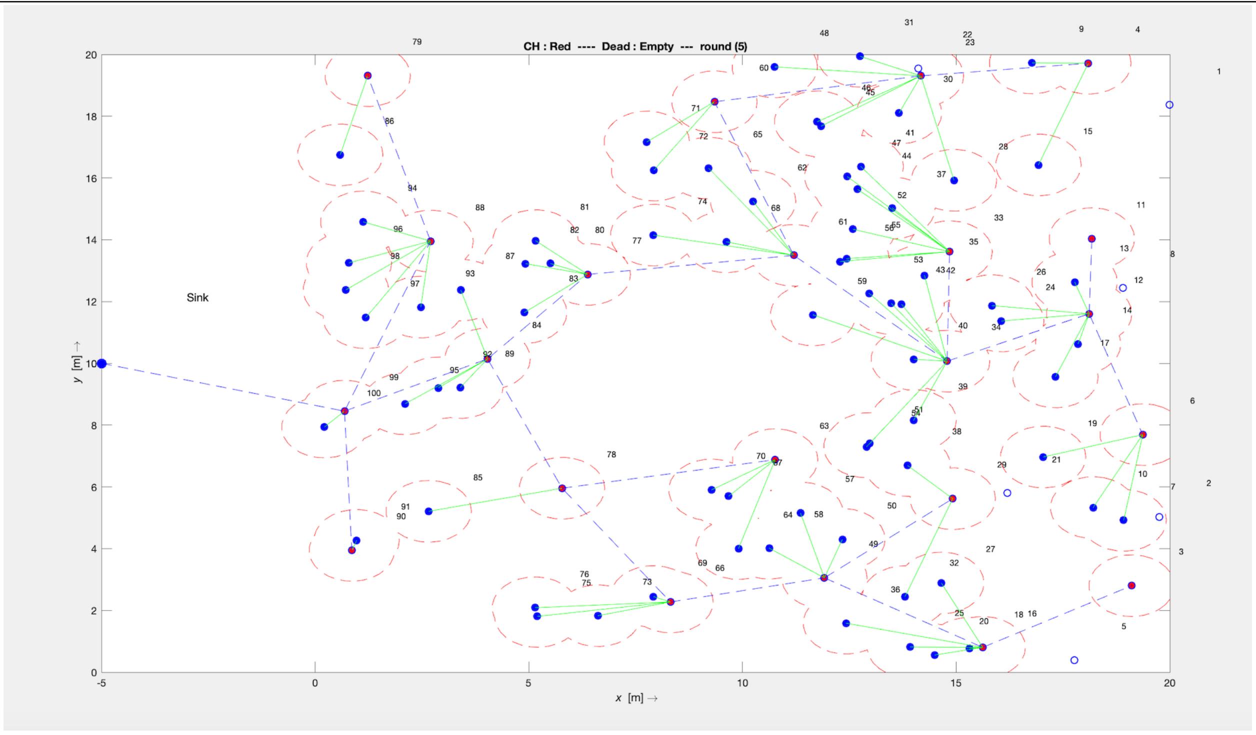


Figure 4.2 Simulation Set-up of HMM Node Scheduling Algorithm.

The simulations were performed over 2000 rounds as the upper pound of the simulation experiment. The breaking point of the simulation experiment e.g., the simulation will completely stop when 95% of the sensor nodes have depleted their energy. In each simulation round, an event is detected and reported to the base station. In this experiment, the performance of the original RCS algorithm against the HMM Node scheduling algorithm version were compared and analysed, in terms of the following metrics: All Used Energy; Lifetime of WSNs; Throughput; Connectivity; Coverage; Coverability. Below are the graphs obtained per metric, in each graph there are two simulation factors:

1. RCS scheduling algorithm, referred to as “scheduling_o”.
2. HMM Node scheduling algorithm, referred to as “scheduling_m”.

4.4.1 All Used Energy

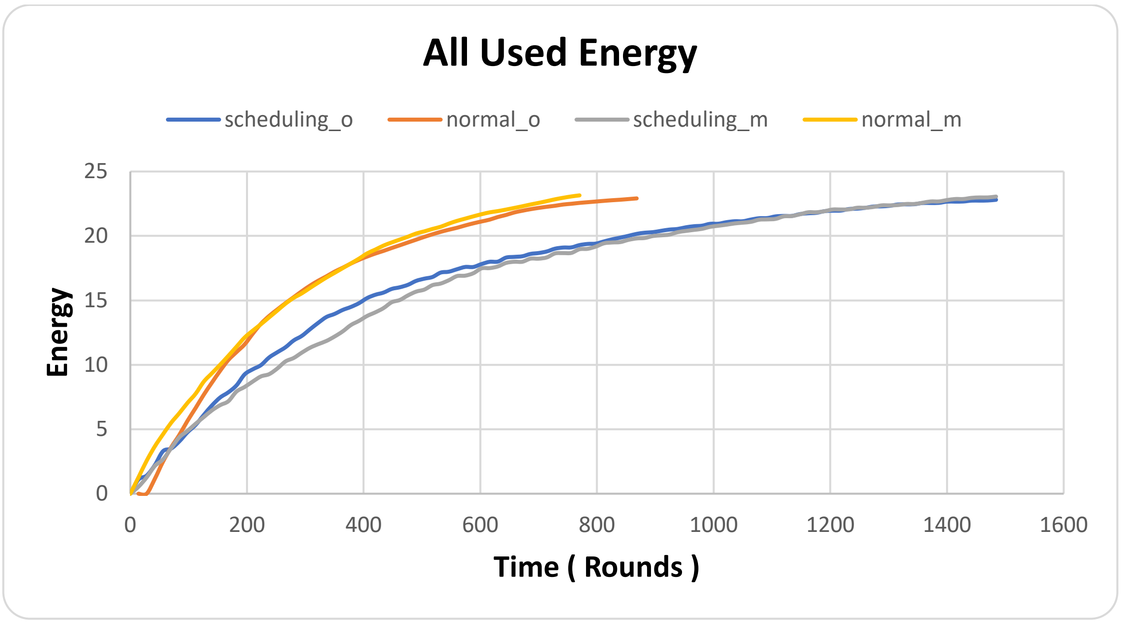


Figure 4.3 All Used Energy

The All Used Energy metric is concerned with the sum of consumed energy in the WSN during its lifetime of operability (simulation time for the purpose of this thesis). The X-axis is the number of simulation seconds/times, while the Y-axis represents the energy consumption unit in Joules as shown in Figure 4.3. The performance of the curves in the above diagram can be interpreted as a metric performance evaluation whereby the less the curvature with respect to Energy levels the better because this shows the WSN achieves the same number of rounds (Time) with less energy. As it can be seen in the Figure 4.3 at the 168th round (Time) the energy consumed by the original RCS is 7.867 Joules, in contrast, the energy consumed by the improved proposed algorithm (using HMM) is 7.133 Joules. Hence, the HMM-assisted algorithm shows improved performance with respect to energy consumption than its RCS peer. This proves the earlier proposed hypothesis in Chapter 1. Furthermore, as time progresses, energy will naturally deplete, and the difference in energy consumption levels between the two algorithms is almost non-existent. This is because there is a positive correlation between the granularity value (of the parameter) and the distance between the curves. This can be noted from round 200th to 600th that the HMM recorded energy saving of almost 2 joules.

4.4.2 Lifetime of Sensor Nodes

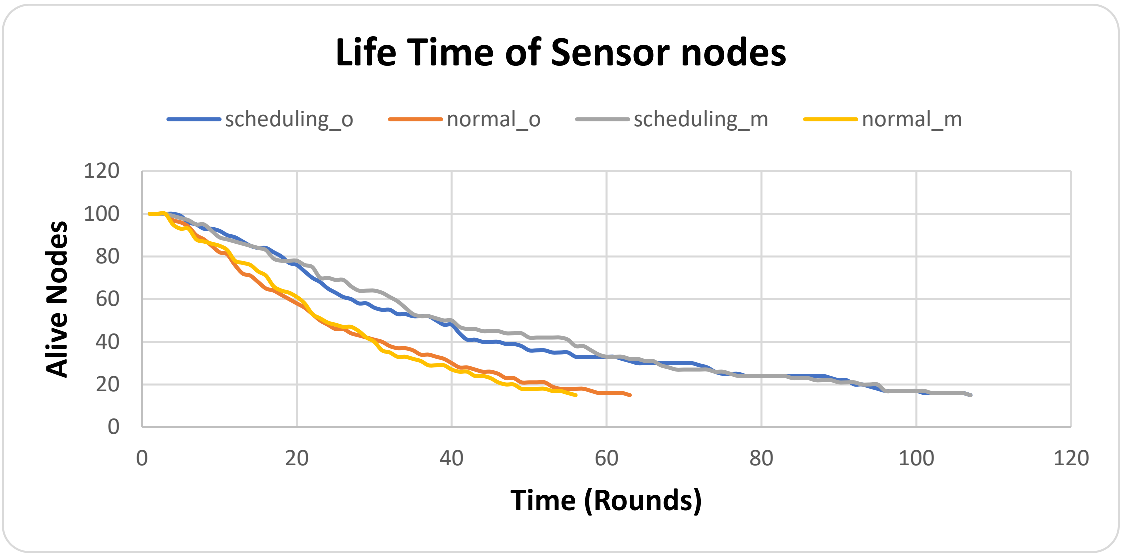


Figure 4.4 Lifetime of Sensor Nodes.

The 'Lifetime of Sensor Nodes' metric refers to how long the sensor nodes will last before their energy deplete. The X-axis is the time in Rounds, while Y-axis is energy of each sensor node as shown in Figure 4.4. From the Y-axis, the remaining number of nodes at the end of the simulation run-time is 15 for both HMM and RCS. Yet, the X-axis shows with HMM the WSN lifetime reaches 107th rounds before the end of simulation, but RCS lasted to 102nd rounds only. Despite the small improvement made with the HMM (of 5 rounds only), this can be explained due to the small value allocated to the sensor parameters (number of nodes, energy, communication, and sensing ranges etc.). Note, assigning large parameter values is an unrealistic experimentation practice [Xianglin et al., 2012]. The differential energies with respect to HMM curve for the 20th rounds to 40th rounds and 40th rounds to 60th rounds are due to the stochastic fluctuations in the energy level for sudden wake up and sleep cycles, these fluctuations are prominent when the number of alive nodes is in between 80 to 40.

4.4.3 Efficiency

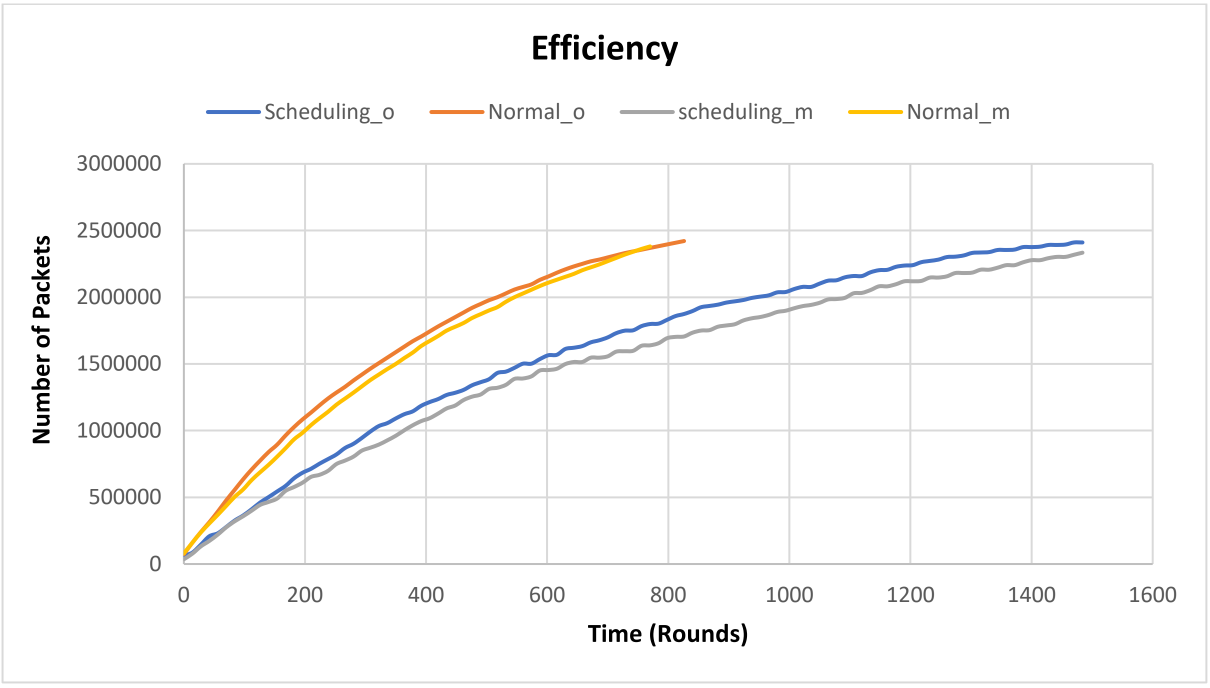


Figure 4.5 Efficiency.

Efficiency is the bandwidth consumed of data packets sent and received in the network. Since the bandwidth of the network is defined as the number of packets generated in the network, hence we can clearly say lower bandwidth will have less congestion and less information. The more bandwidths will result in more congestion and more information. More packets will require more energy for its generation and more packets lost due to congestion which will result in loss of energy and information. The X-axis is the time which is in Rounds, and the Y-axis is the number of data packets transmitted in the networks as shown in Figure 4.5. It can be noticed the distance between the two curves of HMM and RCS algorithms can be explained by the idea that there are lots of packets generated in the networks which indicates that a considerable number of packets are discarded due to collision; the excess generation of packets lead to the wastage of energy and hence, the resultant data transfer is less. With HMM, transferring less packets were possible which resulted in better performance with respect to efficiency. In the above figure, with HMM, the number of packets generated at the 1414th round is 2275957. Meanwhile with RCS, at the same 1414th round, the number of generated packets is 2379214. Hence, HMM generated 10,000 packets less than RCS. Hence there was less congestion in the network and less packet loss and better efficiency than RCS.

Hence, more energy saving leads to a better performance in the WSNs with respect to coverage and connectivity functions. This in turn will provide more available and reliable data readings from WSN.

4.4.4 Connectivity

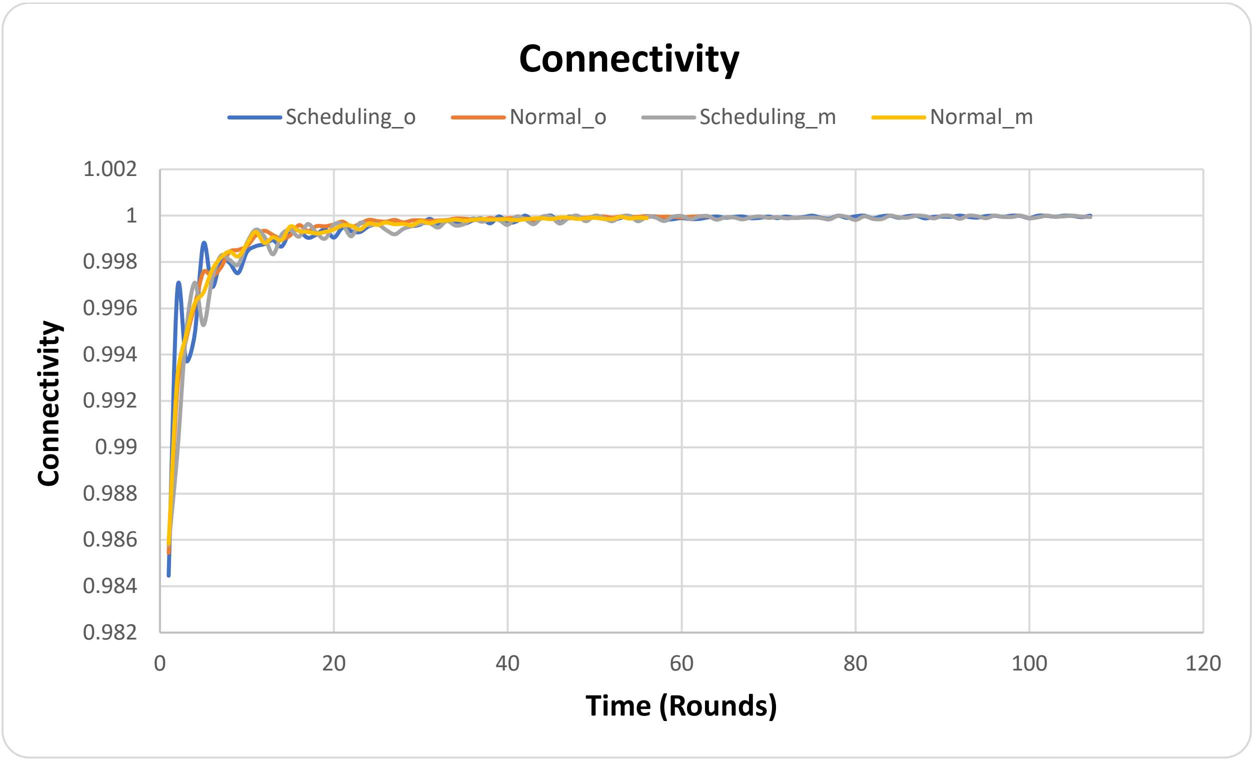


Figure 4.6 Connectivity.

Connectivity is the number of nodes connected at any instance in time. The X- axis is time in rounds, and the Y-axis is the fraction or percentage of nodes being connected. The HMM's connectivity performance is similar to the RCS' performance with respect to the fractions of the nodes that are connected as shown in Figure 4.6. Both algorithms scored equal connectivity values of 99.99, which indicates that there is almost no scope of improvement over the RCS as far as this metric is concerned. The fluctuations can be attributed due to the stochastic nature of the HMM algorithm over the randomization nature of the RCS algorithm.

4.4.5 Coverage

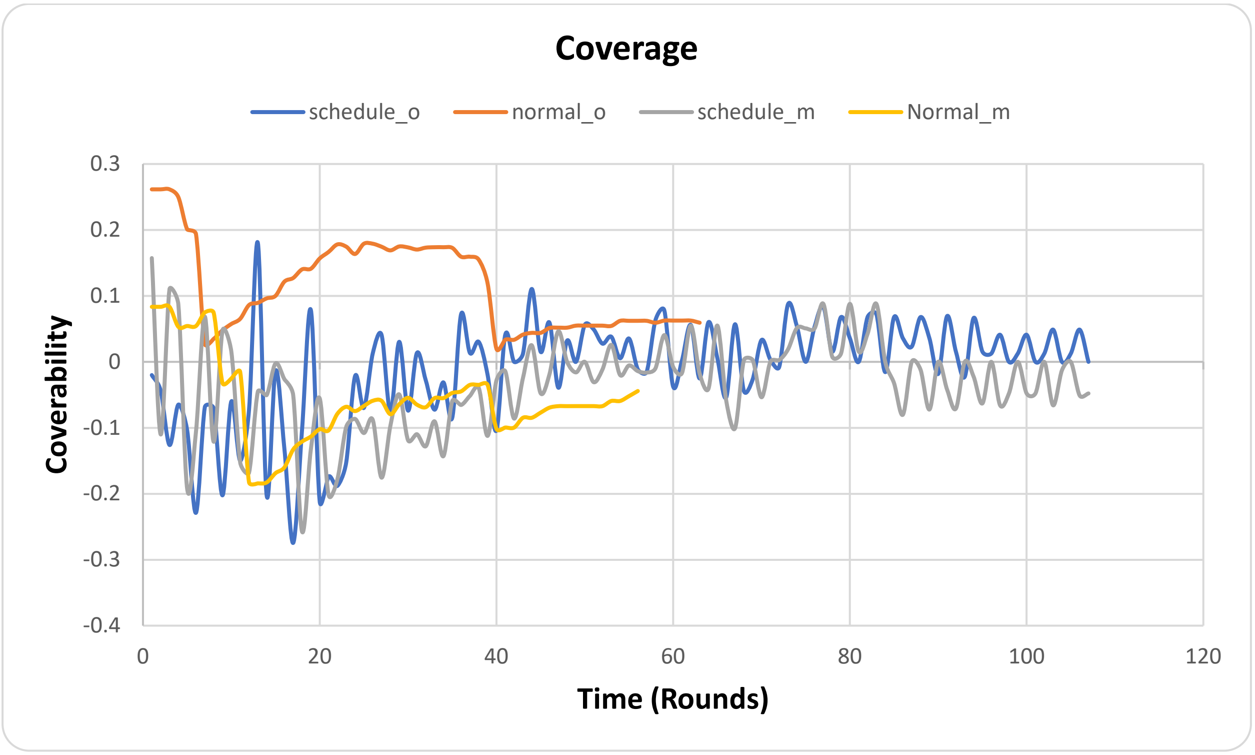


Figure 4.7 Coverage.

The coverage metric is defined as the percentage of the area covered by the WSN. The X-axis is the number of rounds in time, and the Y-axis is the ratio of the coverage for the WSN by large. At the 65th round, for instance, the HMM coverability is found to be 0.005, while the RCS algorithm scored 0.05 as shown in Figure 4.7. This indicates network coverage has been multiplied by 10 using the HMM algorithm.

There is a noticeable fluctuation recorded from the 1st round to 20th rounds, in these rounds, the performance in terms of coverage for HMM is better as compared to RCS because the local maxima obtained in Markov scheduling perfectly managed the decision process in such a way that better coverage can be obtained as seen energy efficiency graphs as discussed above. The curves are the output of a discrete event simulation, so each time the simulation run the output result will have the same result but with the small variation; hence, the curves will fluctuate differently by nature, but the result of the performance remains the same, this is attributed by the fact that discrete event simulation is stochastic by nature.

Another note can be taken from is that from the 20th rounds to 40th rounds is that due to the presence of network holes some negative coverage can be seen in case of HMM but is quickly recovered by adjusting the local maxima in Hidden Markov Chains in sleep, awake transmit, receive cycles.

A remarkable observation is that from 40th to 113th rounds, at these rounds the number of alive sensor nodes is almost 40 %, so now the challenges of keeping the test area covered is equally daunting for both RCS and HMM. In all, the conclusions, the coverage functions are continuously riddled with energy depletion and network hole problems as the number of alive nodes falls below 40%.

The objective here is to study the network's coverage per section of nodes. For 70 nodes out of a total of 120 nodes, the HMM normal coverage (0.31) > RCS normal coverage (0.27869) and HMM scheduling coverage (0.231885) is also greater than RCS scheduling (0.15254) and this trend follows as the number of nodes increases. It is worth noting that the term 'Coverability', while sounding synonymous with 'Coverage', refers to the percentage of the coverage per sensor node. The above figures explain the result of the experiments. The RCS algorithm was originally tested on the basis of coverability, not coverage in C Liu's work [Liu et al., 2006]. This is understandable because the algorithm is not compared or evaluated against other algorithms with respect to (1) the coverage metric of the extra ON nodes, and (2) network lifetime improvement. In the HMM improved algorithm, the aim is for improving network lifetime to contribute to the safety critical system requirements, and further contribute to dependability of such systems. The Coverage metric is concerned with all of the network's nodes, while the Coverability metric is concerned with a certain number of nodes and can be used to analyse single nodes only. Therefore, the Coverage metric has been the focus in the experimentations.

4.5 Evaluation and Interpretation

In this chapter, a new HMM-based node scheduling algorithm is used to address the coverage, connectivity and limited lifetime problem as seen in RCS algorithm. This solution outweighs the computational complexity associated with it by performing the computation of scheduling offline before the actual operation of the network. The main goal is to predetermine the receive and transmit states' schedule of each node in the WSN in the design time. As a result, each node knows its receive and transmit states' schedule and operates accordingly in the runtime. In this process, the transitioning probability between the states have been intervened in that there are receive and transmit states.

For safety aspects, it can be concluded that the proposed HMM algorithm does not lead to safety issues when its scheduling algorithm forces certain nodes to transition to, for instance, ‘receiving’ states. This is mitigated by ensuring both receiving and transmitting nodes at every round of simulation time to ensure network’s operability. The HMM node scheduling algorithms provide a better solution through the utilisation of probability distribution function. This solution is found among many solutions in the state space. Since, the decision is based on information made available by sensor nodes locally. This is an example of localized application of HMM in sensor nodes, and hence the receive and transmit states’ schedule is determined. The limitation here, however, is that this solution is bound to the problem restriction as dictated by WSNs configuration.

The HMM-algorithm is interested only in the energy level of a node and is concerned about two states: transmit and receive. In reality, a node may have other states depending on varied processing it may perform, computing location. These may be represented by an abstract ‘processing state’, however, only the energy level after the processing interests the HMM algorithm. Besides these ‘expected’ states, unexpected events can occur that affect the actual state of a node. A given phenomenon might interfere with the transmission state of a node, turn a node off for a short while then turn it back on, etc. These can be infinite and are modelled by hidden states and affect the probability of transition between transmit and receive states. A limitation to this is that the number of hidden states is undetermined/unknown and has an effect on the performance of the algorithm. Notably, it is possible to model hidden states that never occur in reality; or to have states that occur in reality but are not modelled. In both instances, information about these real states cannot be used by the algorithm to improve its performance. The HMM algorithm is stuck in local minima and local maxima as it proceeds only with information it was given during the initial configuration.

A possible solution might be to insert a feedback loop to provide additional information to the HMM algorithm. This would require monitoring the network to decide what hidden states to add or preserve, which may not be ideal when the WSN is placed in an environment with a lot of dynamic changes. In the next Chapter, a bio-inspired computation to bypass this limitation is considered.

Chapter 5 A Novel Implementation of a Bio-inspired Bat WSN Scheduling Algorithm

5.1 Introduction

This chapter adds to the contributions of a new implementation of a Bio-inspired computation algorithm to solve the MOO problem in WSNs. Since the difficulty with HMM is that it is bounded with suboptimal solutions, the question remains open if further improvement is possible as previous HMM algorithms explore solutions in terms of local minima and local maxima, not in terms of the unique solutions in search space globally. Bio-inspired algorithms are a perfect fit for MOO problems [Singh et al., 2021] which provide the global solutions instead of local values. The proposed Bio-inspired BAT algorithm is performed in two levels: (1) a search space - where a fitness function is created to represent the best solution; and (2) a Pareto optimization technique to organize and then select the best solution in this search space. There are two functions involved with the proposed algorithm, objective, and fitness functions to find the best possible solution, which in this case is the optimal scheduling of nodes' ON/OFF duty cycles.

The concept of the Bat algorithm is based on the use of computational methods inspired by nature. For example, the behaviour of a particular species such as Bats when hibernating (going into a deep sleep) in the cold mountains to preserve energy is an inspiration for the proposed work in this chapter. Recent studies show that certain groups of Bats hibernate by grouping themselves, reducing their temperature, slowing their heartbeat rates, and breathing rates significantly, and expending little energy. When the Bat group decides to wake up, the bat in the middle of the group accelerates its heartbeat, which in turn accelerates its blood circulation, as a result, heat is produced. Finally, the heat is propagated from the closer to the rest of the bats in the group, where they start waking up gradually.

In this work, for the first time, we propose using a Bat algorithm at the application layer to optimize the node scheduling of WSNs along the three requirements of coverage, connectivity, and lifetime. This notably permits the exploration of a larger search space to find the best scheduling solution.

5.2 The Bat Algorithm

In this chapter, the bat-inspired algorithm is first mapped to the WSN model in Figure 5.1 below.

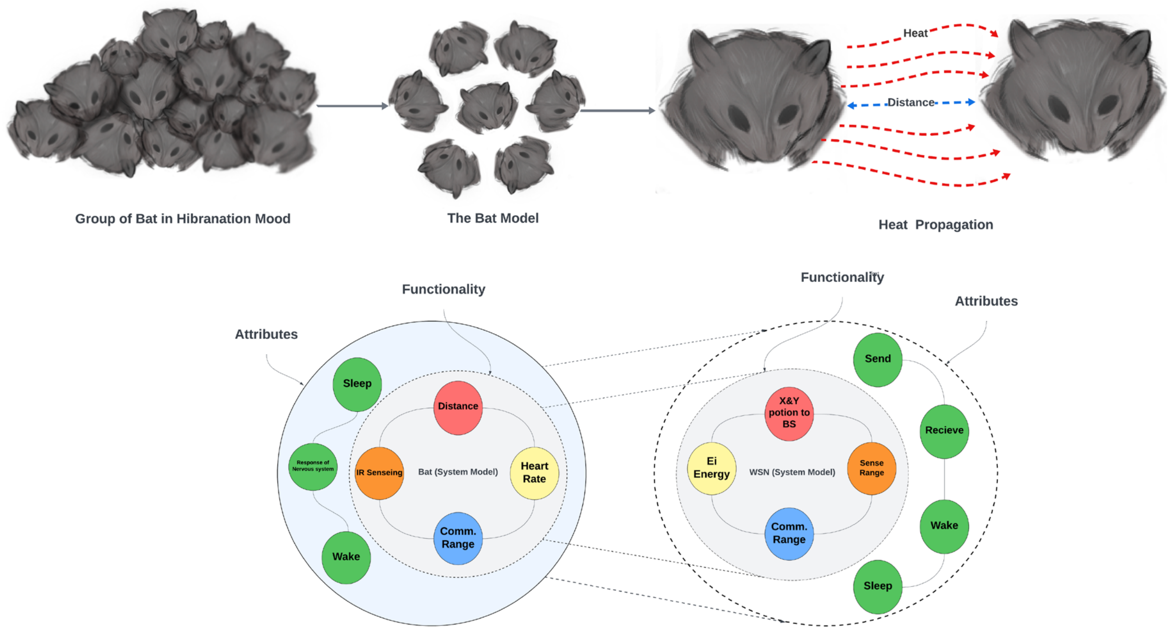


Figure 5.1 Interpretation of the Bat model into the WSNs Model

The Bat-inspired concept here stems from the mechanism bats use where they increase their heartbeat to propagate heat to the farthest bat in the group. The higher the heartbeat of the middle bat in the group, the higher the distance the heat covers. The concept here revolves around two key notions: energy, and distance.

1. Energy: the residual energy of the nodes is catered for. As the network progresses in time, the energy is defined for each node, which will eventually be depleted.
2. Distance: it is the distance of the nodes from the base station. e.g., the X and Y coordinates of each node from the base station.

Below is the pseudo-code developed to describe the functionality of the proposed algorithm.

Algorithm 5.1 represents the pseudo code of the developed approach.

```
Algorithm 5.1 BAT Algorithm
Input: Initial population, frequency, and velocity of Bats
Output: Global umber of iterations.
1: Initialize the Bat population n, the heart pulse rate r(i), the velocity v(i)
2: While (t < max _iter)
3: Fi = f min + (fmax - fmin) * beta
4: Vi (t+1) = vi (t) + (xi(t) - x*)
5: xi(t+1) = xi(t) + vi(t+1)
6:                          If (rand > ri)
7:                          x(new) = x(old) + epsilon* fi
8:                          End If
9:        If (rand < E) && (zi < zsol)
10:       E(t+1) = alpha * E(t)
11:       Z(t+1) = z(t) + b(t)*P(t)
12:       End If
13: Find the solution Zsol for Ei;
14: Rank the Bats in P [],
15. Pareto-optimal (P [])
16:  For each solution s1 in S:
17:        Initialize a Boolean flag dominated = False.
18:      For each solution s2 in S:
19:        If s1 is dominated by s2 (all objectives of s2 are better or equal to s1),
20:        set dominated = True and break.
21:       End If
22:     If dominated is False,
23        add s1 to P as it is a Pareto-optimal solution.
24:    End If
25:       End For
26: Return P as the set of Pareto-optimal solutions.
27:  End For
28: End While
```

The proposed algorithm is inspired by mimicking the mechanisms of how bats operate in energy consumption and network coverage via their heartbeats into the WSN implementations. The framework of the algorithm is depicted and runs with these precalculated input values:

- Location coordinates of nodes
- The energy level of each node at start
- The energy consumption due to Tx and Rx
- The base station location

- The test area dimension
- Time of operation of sensor network
- The threshold value that limits the operation of sensor nodes

These input instances are fed to the proposed algorithm where the following output metrics are obtained.

- Energy values at instantaneous time
- Sleep and awake schedule.

The objective function presents the possible solutions in the search space, whereas the fitness function is a function that analyses a candidate solution to a problem. The candidate solutions are processed as inputs by the fitness function and the produced outputs are analysed by how "fit" i.e., how close the solutions are with respect to solving the problem in consideration.

There are 4 dimensions as input these can be viz. the frequency, velocity, position, and pulse. The difference fmax-fmin is considered as one where f is the max and min value for the Bat population/solution. By using equations 5.1, 5.2, 5.3, 5.4, and 5.5 from the theoretical aspect of bat algorithms, there are updated output values for energy, position, and heart rate with fitness functions as described in equations 5.4 and 5.5. Pareto optimization is utilized over the output values of the bat algorithms to visualize and organize the data in a meaningful manner. The mathematical formulation relates to higher energy levels, which are required to wake the farthest bats. The position can be mathematically determined by adjusting the energy levels. The novel approach proposed in this paper stems from adding convergence time and energy to the multicore space search problem.

In the search space, there are solutions, and, in the solutions space, there are all the solutions listed among the solutions, there is one best solution. Let the coordinates of a Bat under consideration in search space be (Z_i, E_i) where Z_i is the position with respect to time (position with reference to base station) and E is the energy of the Bat with respect to time. The distance between nodes and the base station is also a contributing factor to the search space, the general structure of the Bat algorithm is borrowed from [Yang, 2020]. Therefore, the search space equations are represented as:

$$Z[t+1] = Z[t] + E[t+1] \quad (5.1)$$

$$E[t+1] = E[t] + [Z[t] - k[t]] \quad (5.2)$$

Now, the solution is obtained in the search space, we need to make sure these solutions are withing certain range/loudness (viz., the loudness is related to the amplitude of the wave generated by bats. The factor C is used to bring the loudness within the specific range). Unlike the standard bat algorithm which uses echolocation to generate loudness, the modified Hibernation bat algorithm uses the Heat rate to generate waves, so this heart rate has an amplitude as it has loudness.

$$A = fmin + [fmax - fmin] * C \quad (5.3)$$

$$C = [A - fmin] / [fmax - fmin] \quad (5.4)$$

where f is the heart rate of the bat under consideration, whereas C is a gradient (steepness/ slope of a curve in search space) average position. Where k(t) is a proposed global solution, not possible as the energy is depleted with respect to time because the sensor battery is drained after each cycle. Hence, the solution proposed in this paper will be local. This algorithm will have both local and global solutions. So, the local solution is:

$$Z[t+1] = Z[t] + b[t] * P[t] \quad (5.5)$$

Where P[t] is the population density of Bats and b can be a varying constant. P(t) is a normal distribution of the bats with high density inside and sparse outside. In other words, The Bat population are highly concentrated/dense as you get close to the centre and spares as you move out away from the centre of the bat cluster. The global solution can be obtained by the addition of the P[t] variable to the local solution as described. If the population P[t] can be averaged out by the following relation representing the global solution by

$$P[t+1] = c * P[t] \quad (5.6)$$

This equation describes the average number of bats per layer (hops in terms of a sensor network in the cluster, to have an average of the bats per layer, equation 5.6 is used) this population variation equation with respect to time. This triggers multiple solutions rather than a single solution.

In this thesis, the search space is in the remit of the energy of sensor nodes and how the energy depletes with respect to the network's performance. Assuming the following nodes-schedule as proposed in the following Table5.2, where each node will have various On-Off time instants.

Table 5.1 Nodes scheduling with variant ON/OFF time instants.

Nodes ID	**Instant 1**	**Instant 2**	**Instant 3**	**Instant 4**
Node 1	OFF	ON	OFF	ON
Node 2	ON	OFF	ON	OFF
Node 3	OFF	OFF	ON	ON
Node 4	ON	ON	OFF	OFF

If time values are arbitrarily assigned to the instants such as: Instant 1 = 1; Instant 2 = 2; Instant 3 = 4; Instant 4 = 7, but applying some arithmetic's:

- Instant 2 - Instant 1 = 2 seconds - 1 second = 1 second;
- Instant 3 - Instant 2 = 4 seconds - 2 seconds = 2 seconds;
- Instant 4 - Instant 3 = 7 seconds - 4 seconds = 3 seconds;

The proposed algorithm processes the residual energy of bats/nodes into a function of On-Off values with respect to time. Since energy depletes as time naturally progresses of the WSN, the residual energy is a proportional attribute to time, the On-Off values will be dynamically updated from the proposed algorithm. The premise of the work is based upon a previously calculated data table which contains the sleep awake and energy, communication requirements, scheduling values. The core of the proposed algorithm is to optimize the objective function related to the product of energy, distance, and network lifetime as explained in the following scenario 1. In the proposed solution, the objective function is considered which takes into its optimisation goals energy (drive the network and lifetime), distance (related to connectivity), and network lifetime (related scheduling).

Use-Case Scenario of the BAT Node Scheduling Algorithm

The following is a step-by-step use-case scenario based on three Bat populations presented to demonstrate the mechanism of how the Bat algorithm provides a solution for the problem at hand. The BAT algorithm considers in its search space an objective function and a fitness function to achieve the best optimal solution. The objective function is the function where it contains the main parameters which represent the solution in the search space, the main aim is to produce the best scheduling solution. The objective function is represented as:

Objective function= f (frequency, velocity, position, pulse)

Where its primary parameters define as:

- Frequency: The Rate of ON and Off nodes in the WSN.
- Velocity: Is the mobility status of our node in our scenario it is 0 as our Bat (Sensor Node) are not moving stationary.
- Position: Is the X and Y coordinate position of the sensor nodes or Bat
- Pulse: is the residual energy of the node.

1: Initialize the bat population, energy levels, and the objective functions;

The first step is where the population of sensor nodes is initialized with the energy value and an objective function randomly as in Table 5.3.

2: Define heartbeat rate (relates with the energy), for the bat population sample (number of bats), population density (number of bats/total test area)

The second step heartbeat rate is defined (relates with the energy), for the Bat population sample (number of bats), population density (number of bats/total test area); the heart Pulse rate is the energy level of the sensor node (Bat population is number of sensor nodes). For example, suppose there are 100 Bats, and the test area is 100 square meters then the population density is 100/100 =1 Bat per square meter (coverage).

Table 5.3 is the 1st iteration with random values assigned to the variables. In this step the initialization take place.

Table 5.2 Initializing the population of Bat with random values

Bat 1	Bat 2	Bat3	Variables	C threshold value
10%	15%	13%	Frequency F (percentage or population of Nodes ON)	F > 0
0	0	0	Velocity V (mobility status)	V = 0
1	3	5	Position N (No. of hops)	N < Number of nodes
25	26	30	Heart Pulse E (energy)	E <= 30 J

3: Generate new solution by adjusting the heart rate, and updating the energy level with locations to generate the new solutions;

4: $(if\ ((rand > c = true;)\quad // c\ = threshold\ value;$

Condition: There is a lower threshold values for each variables Frequency > 0, velocity =0, no of hops < number of nodes, energy level <= max energy of the battery used in sensor nodes. From the above, Bat 2 is selected amongst the three bat populations because it has the highest frequency population with a moderate energy level good then this condition makes it suitable to have better connectivity and coverage.

5: Choose the best solutions based on the fitness function defined by equation 5.5;

The fitness function operates as follows:

Fitness function: z(t+1) = z(t) + b(t)* p(t)

Where t is the seconds t=1 seconds and b is b=0.02 similar to the learning parameters.
z (2) = z (1) + b (1) *p (1)
= 10 + 0.02*10 =10(1+0.02) = 10.2 Bat 1 (1st population)
= 15(1+b) =15(1+0.02) =15.3 Bat 2 (2nd population)

= 13(1+b) =13(1+0.02) =13.26 Bat3 (3rd population)

6: else go step 4;

Step 6 means (Else go to step 4) means if you arrive at all the same values for all bat populations, it will execute this step to recalculate from the beginning.

7: if (rand) > A && (zi < z solution) = true;

This step A means selecting those Bats which have the lower energy values to achieve these objectives (the minimum operating energy of the Sensor nodes). The objective here is to achieve 100 % efficiency in utilisation, which is the ideal Z solution. For example, suppose a network requires 3 nodes with minimal operating energy and two sensor nodes have met the condition while one has not. In that case the energy for the 3rd node must tune up to join in.

8: update the solution. Increase Ei and reduce Zi;

Step 8 updates the solution, where the energy level Ei is increased and the Zi is reduced. This means the more energy/node fewer nodes are involved and the less energy/node more senor nodes are involved.

9: Rank the bats, find the solution Zsol for Ei;

At this step after the Bat 2 solution is accepted, and ranking of the best to worst solution is taken. Ei is reset and Zi is reduced. There is a solution generated after each iteration. This iteration is based on the Bat population. Finally, the best solution is communicated to the base station.

5.3 Experiment Assumptions

In this work, a uniform deployment with a random distribution over an area to be monitored was considered. It is assumed that the area to be monitored is composed of multiple grids, where each grid represents the Points Of Interest (POI) [Liu et al., 2006]. In order to ensure that the event is detected and reported back to the base station, we assumed that the network coverage intensity was high and was based on the probability coverage model [Sendra et al., 2015]. Thus, a POI is defined as the area to be covered by at least one sensor node in the WSN.

Moreover, to ensure a good level of coverage, the coverage metric is defined to be greater than or equal to the area to be monitored. Consequently, the assumption is made that when connectivity between sensor nodes is established, all nodes are connected in the WSN. Thereby, there is always a maintained path from the base station to all nodes deployed over the area to be monitored. Furthermore, the network lifetime is defined as the most useful lifetime where sensor nodes maintain a level of connected coverage with the expected network operational lifetime over the area to be monitored. Figure 5.2 represents the possible spatial coverage scenario over the area to be monitored. In each scenario, the level of node density distributed randomly within a certain zone can be noted.

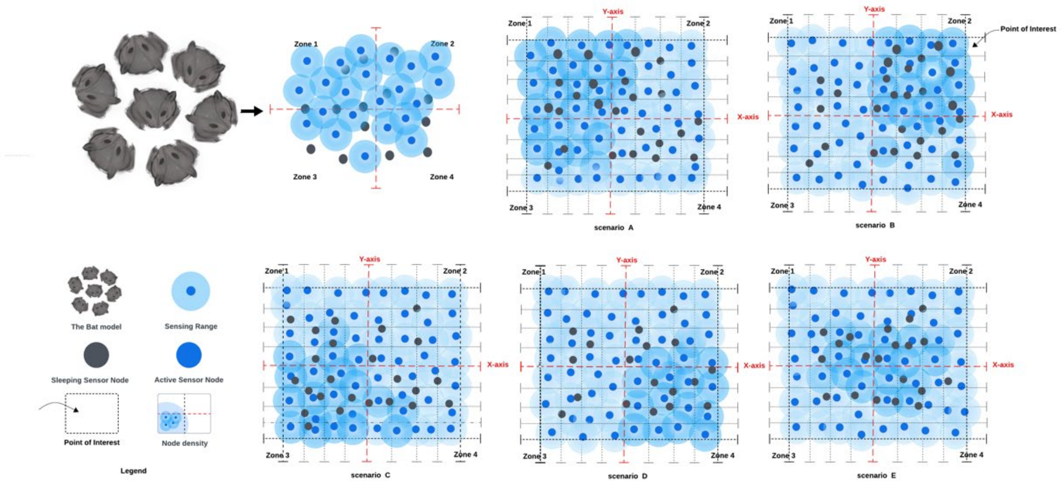


Figure 5.2 Spatial Connected Coverage Scenarios (A, B, C, D, and E)

5.4 Experimentations Set-Up

MATLAB-based event-driven programming simulator is used in this work. The simulation is performed to study and analyse the performance of the new proposed bat algorithm and compare it against the HMM algorithm, introduced in Chapter 4, and a normal non-scheduling, cluster-head, algorithm. In this simulation, a 20 m^2 of an area is to be monitored. The simulation is composed of rounds, in each round an activity of sensing and reporting the sensed data back to the base-station. The simulation parameters are identified Table 5.4 below:

Table 5.3 Simulation Parameters

Parameter name	**Value**
Length of the network (meter)	W = 20
Width of the network (meter)	L = 20
Initial energy of each node (Joules)	Ei = 0.25
Packet size for cluster head per round (bits)	CHpl = 3000
Desired percentage of cluster heads	p = 5/100
Max Number of simulated rounds	num_rounds = 2000
Average Time in seconds taken in setup phase	Tsetup = 4
Average Time in seconds taken in steady state phase	Tss = 10
Energy for transmitting one-bit	Etrans = 1.0000e-09
Energy for receiving one bit	Erec = 1.0000e-09
Data aggregation energy	Eagg = 1.0000e-09
Energy of free space model amplifier	Efs = 5.00e-8
Distance from the sink to sensors area	Marg = 5
Packet size for normal node per round (bits) for each type	rates = 80
Max range for wireless transmission for each node	R = 3.5

The simulation is composed of 2000 rounds with a threshold to stop when the last number of nodes consumes all its energy. The Figure 5.3 below illustrates the experimentation set-up with the area to be monitored using 150 nodes.

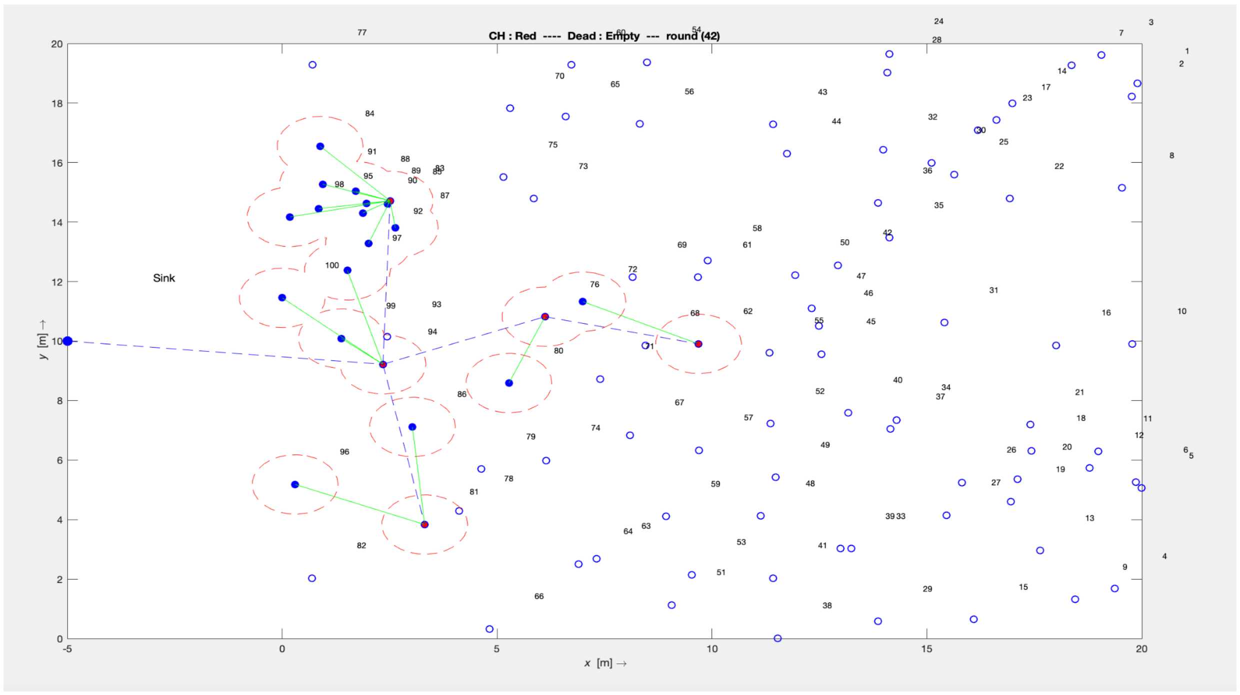


Figure 5.3 Simulation Set-Up of the Bat Node Scheduling Algorithm

Simulation was individually tested for 30 times. 1-way ANOVA analysis was implemented on BAT and HMM algorithms to identify the p-value for statistical results.

Table 5.4 Statistical Analysis

Metrics	BAT Algorithm	HMM Algorithm
Number of Hops	26	20
Connectivity	0.99	0.99
Coverage	0.77	0.56
Lifetime	196	183
ANOVA 1(P-value)	4.12002e-188	3.34456e-148

The P-value in table 5.5 represents an observation for the 30 times run test. If the P-value is less than 0.05, this means one or more values among the tested data are greater than the assumed mean. Thus, rejecting the null hypothesis, otherwise accepting the null hypothesis; implying no extreme difference between the values. In this research, the P-value is below 0.05 as shown in Table 5.5.

5.4.1 All Used Energy

The Y-axis is the nodes' used energy, and the X-axis is the round number. The energy usage for the bat algorithm is better as compared with the RCS and HMM algorithms.

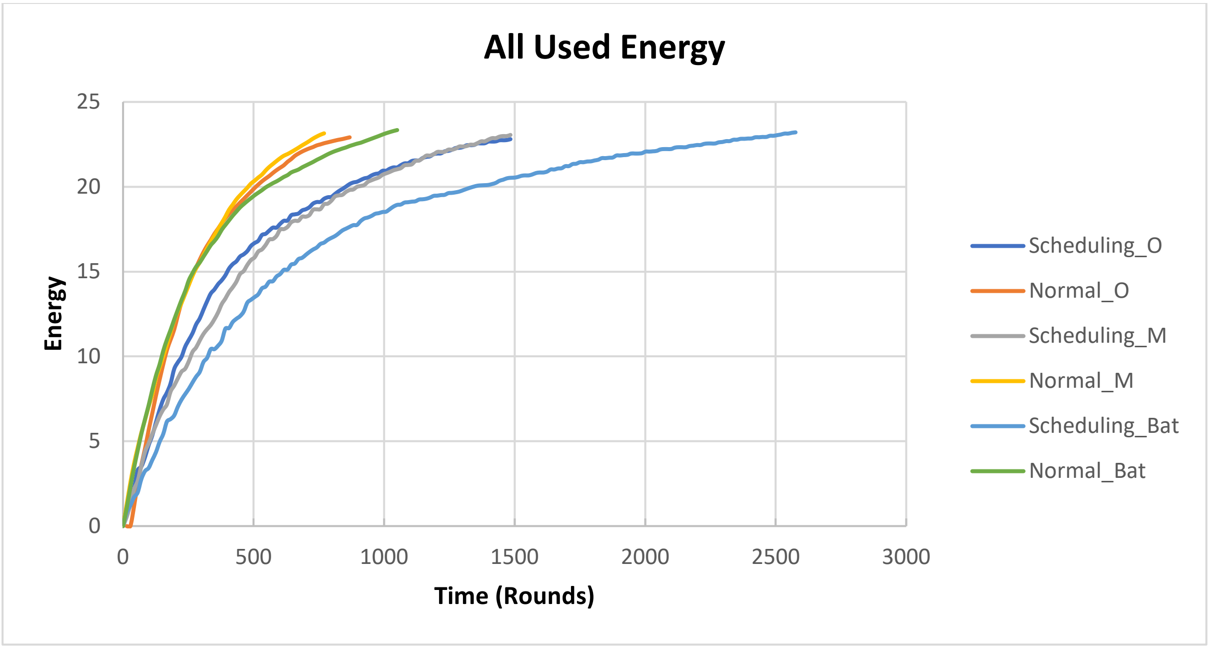


Figure 5.4 All Used Energy

It can be noticed that the BAT algorithm reached an energy consumption of 24 joules at the 2555th round, while the HMM algorithm scored it at the 1000th round. This shows how energy efficient the Bat algorithm is by preserving energy for more than the double of time rounds which was achieved by the HMM. As energy is one of the main critical parameters so using it cautiously is one of the prime requirements of the WSN. As described earlier the Bat algorithm uses energy as one of the optimizing parameters due to the stable energy considerations in terms of distance as depicted in Figure 5.4. However, in the HMM and RCS, there was randomisation of node allocations in the set-up phase to join WSN subsets, hence their ON/OFF deployment will be based on that result. Additionally, there were extra-ON nodes to fill the network's coverage holes. The main advantage that the BAT algorithm's success over HMM is the ability to find global solutions more efficiently than HMM, though HMM 's successful ability to find a set containing global as well as local solutions. Hence, we can see the abrupt performance hike in the case of BAT without expending more energy. Preserving more energy reflects automatically on the performance of the sensor nodes in terms of coverage and connectivity. Hence, providing a better performance to obtain a better or optimal QoS of the WSN.

When not considering the sensor residual energy, the performance was compromised with respect to RCS and HMM. This result substantiates that the residual energy information is useful in extending the time taken to the first depletion of node energy in the Bat Algorithm.

5.4.2 Lifetime of Sensor Nodes

The Y-axis represents the number of nodes alive, and the X-axis represents the number of rounds. The BAT algorithm is improved noticeably in terms of nodes' lifetime energy as can be seen in Figure 5.5. As can be noticed in the BAT algorithm the number of nodes that stay alive are 18 nodes at the 181st round. It can be noticed that the BAT algorithm outperformed the HMM and the RCS algorithms by a noticeable margin.

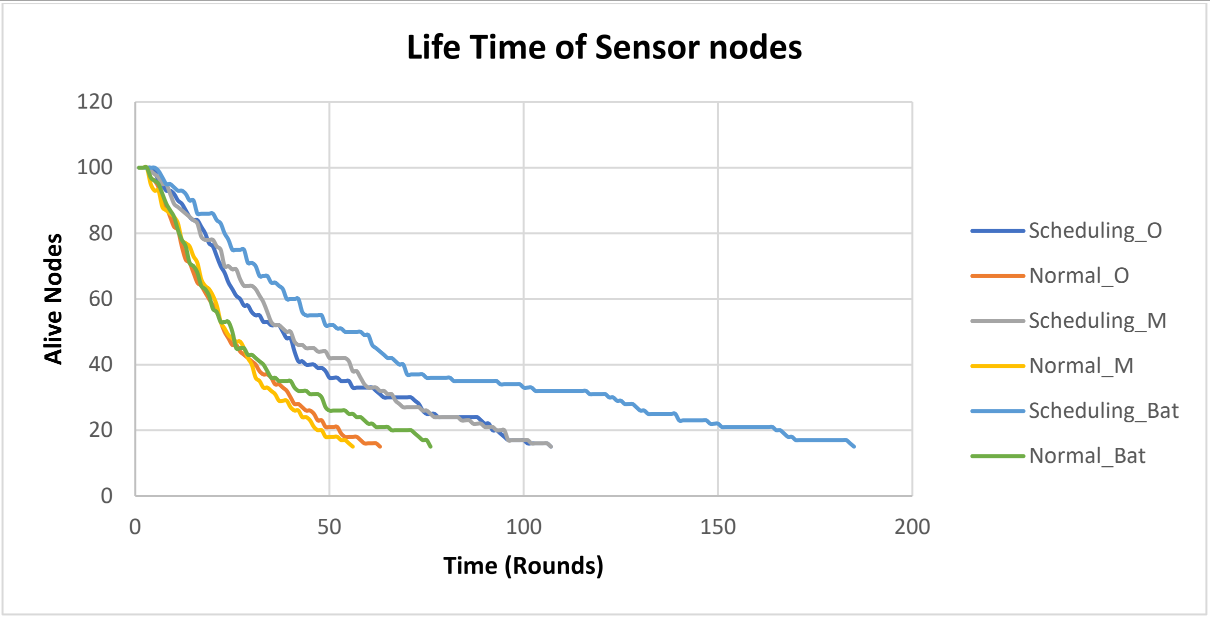


Figure 5.5 Lifetime of Sensor Nodes

The RCS algorithm uses a random coverage model utilizing an integrated method that provides statistical sensing coverage and guaranteed network connectivity. However, the network's lifetime was compromised in this experiment due to the randomisation of allocating nodes in joining network subgroups. Furthermore, there were nodes with extra-ON states to compensate for the coverage holes. This all results in energy consumption limited efficiency to a limited time period of the WSN lifetime, then eventually there will be energy consumption inefficiencies across the whole WSN for the remainder of its lifetime. Initially, we can notice that the Bat algorithm as well as the other algorithms has recorded a sharp decrease from the 1st round up to the 60th round there is a sharp decrease in the curves due to the random deployment of the nodes and loss of energy due to setting up the network. However, the bat has retained in the number of nodes a life until the 190th round this is due to the optimal global solution as provided by the BAT algorithm.

All these lead to saving up energy in using superior global solutions, hence we can use the utilizable energy in performing other functions like connectivity and converge and lifetime efficiently

The proposed Bat algorithm here solves these shortcomings by replacing the nodes allocation randomisation with an objective function focussing on network lifetime, coverage, and connectivity during the network's runtime. This objective function introduces pre-determined schedules for the nodes across their lifetime, enabling the nodes to be fully aware of their ON/OFF states without disturbances due to randomisations. This determinism in stochastic scheduling improves the network's lifetime, coverage, and connectivity. These schedules identify the WSN's optimal points as illustrated in Figure 5.3.

It can easily be found in Figure 5.4 that those optimal points of energy and distance considerations were on the upper part of the curve or somewhat less scattered. Furthermore, when the bat algorithm was implemented, coupled with pareto, it provided those points in the space which were noticed to be stable. The proposed approach here allowed us to sustain the process of dissemination of messages over the network for a longer time.

5.4.3 Efficiency

The Y-axis represents the number of packets sent and received, and the X-axis is the number of rounds.

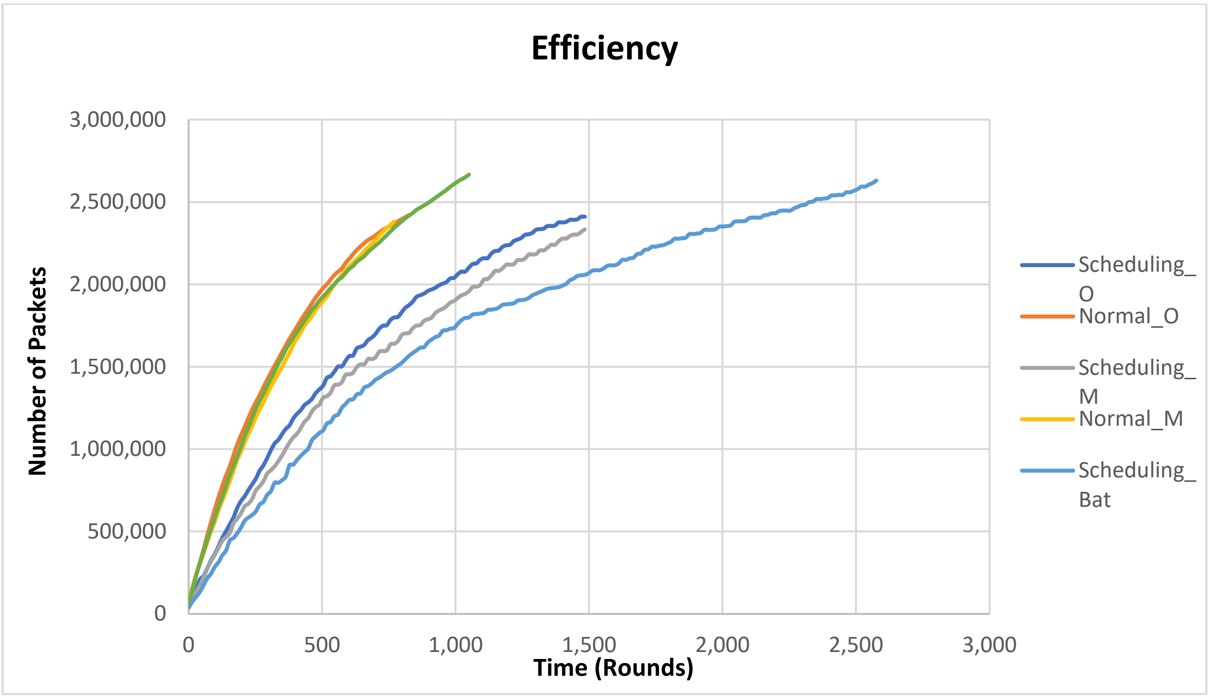


Figure 5.6 Efficiency.

During the lifetime of the networks there are a number of packets sent and received, and the throughput of all the mentioned algorithms were compared with the proposed Bat algorithm. The correlation measured here shows that the longer the network lifetime the higher the packet send/receive rate is. It is noticeable from Figure 5.6 that the Bat algorithm has recorded a higher throughput than the other algorithms. For example, the bat recorded 2,500,000 packets of throughput at the longest lifetime in of the recorded simulation at the 2,555th round.

Since the bandwidth of the network is defined as the number of packets generated in the network, it is clear that lower bandwidth will have less congestion and less information. and more bandwidths will result in more congestion and more information.

Hence, it can be noted from the graphs that the BAT algorithm provides the trade-offs between congestion and information. Hence the efficiency of the network will be better as compared to others. It can be noticed that the Bat algorithm has recorded the lowest packet generated (less congestions) from the 1st to 1000th rounds.

The more packets generated will use the maximum bandwidth available and hence the energy will be lost due to excess packet generation and hence other functions like connectivity coverage and lifetime will be hampered.

5.4.4 Connectivity

The Y-axis is the performance of connectivity which is measured by number as a value fraction of nodes connected in the network. Here there are 150 nodes in the network, if the connectivity is 0.98, then the total connected nodes are 150*0.98=149. The X-axis is the number of rounds.

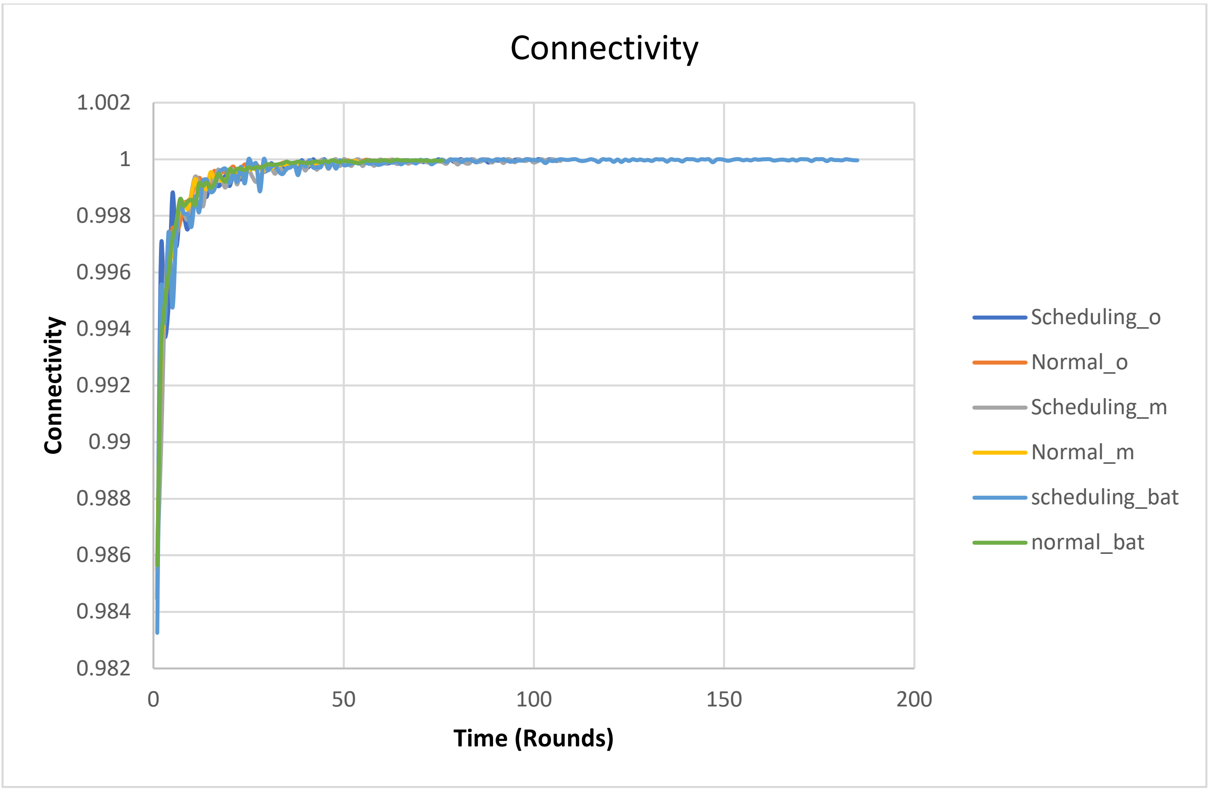


Figure 5.7 Connectivity.

As can be noticed from Figure 5.7, at the 140th round to 158th round the Bat algorithm recorded a connectivity value of 1 which means that the network is live, and it has better connectivity in comparison to the other algorithms. This observation shows the bat algorithm has achieved the highest connectivity measurement with respect to the number of bat-optimized energy (E) and the distance (d) manifested in the prolonged network lifetime. The network lifetime was defined as the time slots passed until there was no set of active nodes that covered all POI or guaranteed the connectivity between the sink and sensing nodes. As E and d reached the stable point as shown in Figure 5.7, the average lifetime has also increased.

Again, the Bat algorithm has recorded that from 1st round to the 30th round starts a higher a higher number of nodes are connected due to the random deployment of the nodes and loss of energy due to setting up the network. the remaining part is stabilized due to the algorithm providing general global solutions.

5.4.5 Coverage

In Figure 5.8, the Y-axis is the ratio of the coverage to the total area to be monitored, and the X-axis is the number of rounds.

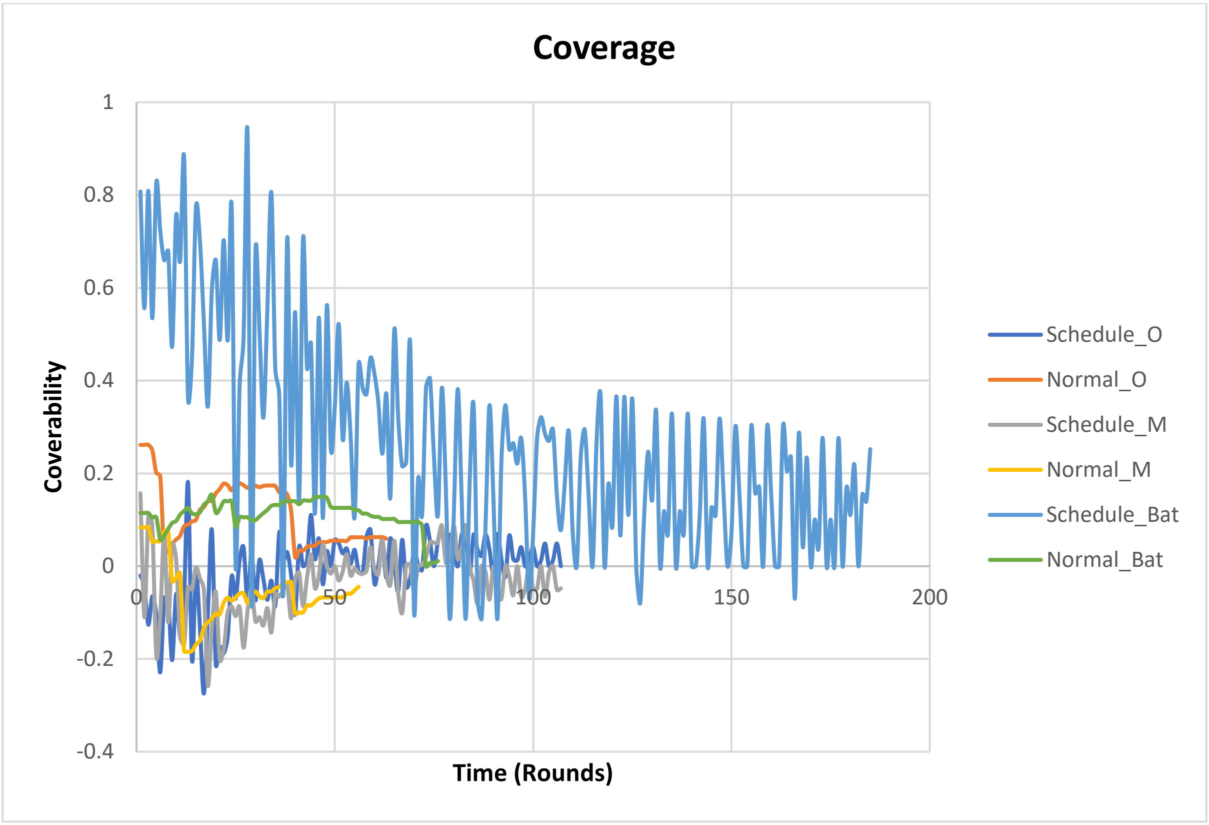


Figure 5.8 Coverage.

As can be noted the BAT has an improved coverage at the 180th round. The HMM recorded coverage of 0 at round 100 because the nodes there have already lost their coverage since they prematurely depleted their energy reserve. In case of coverage the fraction of the area covered by the network. If the area of the test field is 100 m2, the coverage of 0.8 would mean 0.8*100 = 80 m2 has been covered. More nodes in the network are presented in Figure 5.8 from the 1st to 75th rounds in the simulated experiment and hence better coverage is demonstrated. In addition, it can be noted that the Bat algorithm coverage stabilization from the 75th round this is due to nodes' energy consumption in the early setup phase of the lifetime of the bat algorithm, hence providing the global solution.

5.5 Evaluation and Interpretation

To improve the coverage, connectivity, and network lifetime of WSNs, many scheduling algorithms are possible. However, how good an algorithm is, or if an algorithm is the best is not trivial to answer. Bio-inspired computation opened the way for search space exploration; hence, a Bat-based scheduling algorithm has proposed to determine if a solution better than that proposed by an earlier HMM algorithm is possible.

The novel, proposed Bat algorithm has improved the network lifetime and has in turn improved the service availability for more dependable WSNs. The Bio-inspired Bat-based node scheduling algorithm prevents the languishing of the values within the local maximum and local minimum. Instead, it provides stable global maximum and minimum values which remain the same throughout the timeline of the algorithm. In particular, the improvement of network coverage, connectivity and network lifetime were considered as can be seen in the implementation Section 5.3. In addition, from the algorithmic point of view, the Bat and the Pareto optimization algorithms worked in tandem to improve the performance of the network parameters like connectivity, coverage, and throughput. The network lifetime, coverage, and connectivity were improved in contribution to the dependability of WSNs where assumptions/simulation parameters were used to make the work presented here more relevant to real-time WSN scenarios. The specification of the Bat algorithm improves energy and distance which were not accommodated in other algorithms experimentations where distance is interpreted as connectivity, and energy is referred to as lifetime.

While programming in a MATLAB environment was possible, the handling of discrete variables with MATLAB's own optimization technique uses mixed integer optimization which can sometimes be non-trivial. For example, the solution can sometimes be wrong. This is due to the approximation used in mixed-integer optimization techniques. This was solved by manually implementing, coding, the whole Bat-based node scheduling algorithm without using MATLAB functions.

The Bat node scheduling algorithm can be compared and evaluated with other algorithms like the HMM or RCS. In our case, the new proposed Bat node scheduling algorithm has been compared with the HMM scheduling algorithm as it is the latest improved version. Our results show significant improvements on all three dimensions, and Pareto sorting guarantees us that the proposed Bat node scheduling is the best, for all possible fitness and objective functions and for a given search space. This leaves open the possibility for another algorithm to outperform our Bat algorithm. Hence our quest for finding the best scheduling algorithm for a WSN is not over yet. Therefore, in the next chapter, we provide a new representation of the MOO problem based on the Unsupervised Machine Learning Model utilising the SOFM.

Chapter 6 A New WSN Scheduling Improvement via the Self-Organizing Feature Map (SOFM)

6.1 Introduction

This chapter explores the state space of the solution domain from a spatial perspective based on an Unsupervised Machine learning approach. The new approach utilises the SOFM model and proposes a new node scheduling algorithm to enhance the dependability of WSNs by finding the best optimal solution for the MOO problems. Therefore, spatial-based representations of the solution space are explored to obtain the right combination of nodes for efficient scheduling which extends the needed level of availability and reliability of the network service, as identified in [Chapter 1]. The Kohonen map or SOM is a technique that allows dimensionality reduction for the data in the state space from a higher dimensional space to a lower dimension [Taleb, 2021]. The SOM technique has a unique property of producing feature maps that can be visually observed and then realized; hence it is known as SOFM. By proof, the feature map of sur-faces tends to converge on global maxima and minima rather than local maxima and minima. In SOFM, nodes are programmed to learn and become their input values/parameters by mapping their weights to follow their given input data [Sharma et al., 2011].

The SOFMs learn through their unsupervised, yet driven, learning. SOFM nodes are programmed to learn and become their input values/parameters by mapping their weights to follow their given input data [Sharma et al., 2011]. To illustrate the principle, an example is drawn here as the basic idea is to classify and identify datasets, for example, if there is a basket full of green apples and red apples. One can write an ML algorithm to classify them with respect to their colour. For ANN, it can even cluster (group) with SOFM. Clustering can be based on colour or weight. In clustering there is a feature associated with it, if colour is used, the red apples and the green apples are separated. If weight is considered like the apples which have a weight less than 250g to be considered in a cluster all kinds of apples will be placed considering their weight.

6.2 SOFM Overview

A typical WSN is formulated out of multiple sensor nodes connected to each other through one or two base stations to provide application services [Sohraby et al., 2007]. The WSN applications can range from simple detection systems, i.e., environmental monitoring systems, to complex safety-critical systems, i.e. forced fire detection systems as shown in Figure 6.1. Such safety-critical applications have real-time requirements which in some cases require an immediate alert response from the network [Rausand, 2014]. For example, a forest fire detection system requires rapid detection of a fire breakout, so the incident can be mitigated as soon as possible without causing any casualties such as loss of forest, animals, and people in the forest. Another example is the use of the Earthquake Early Warning Systems (EEWS) requires instantaneous detection of the ground motion before, during, and after the earthquake [Sohraby et al., 2007] [Knight, 2002] [Aslan et al., 2012]. Therefore, the specification requirements of a dependable WSN safety-critical system must be taken into consideration to ensure proper and functional operations.

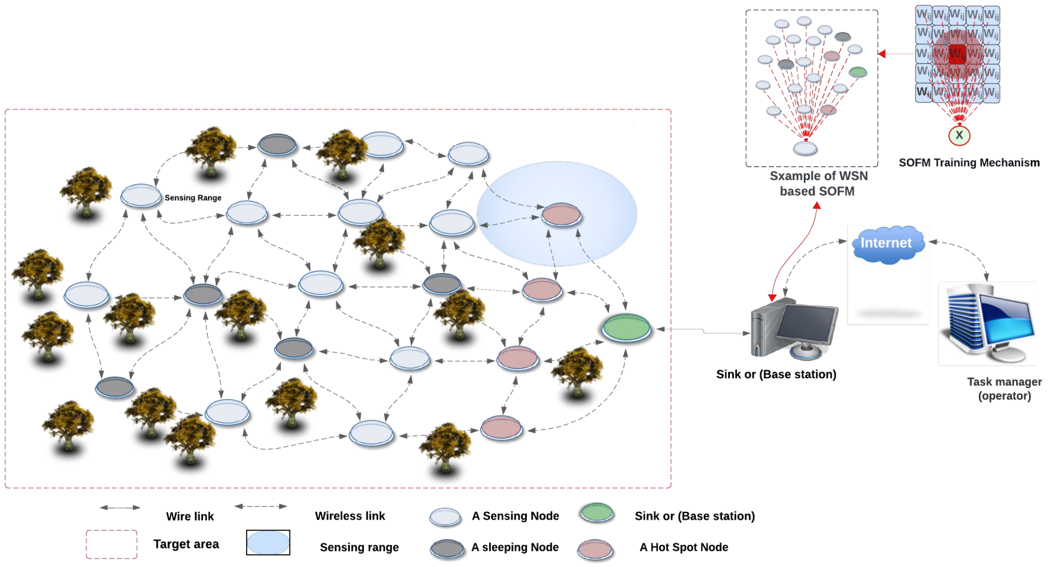


Figure 6.1 The WSN Architecture with the use of fire forest detection systems.

In principle, SOFM is the process of randomly choosing a node from a Map or a Grid and comparing its distance with the distance of all other nodes in the Grid. Then a comparison is performed by calculating the difference in distance between the chosen node and the rest of the nodes in the Grid. There are many methods to calculate the distance between the nodes in the

Grid. The Euclidean distance is one method; which is the shortest line between two points A source and B destination as shown in Figure 6.2 [Mittal and Kumar, 2015].

For example, in Figure 6.2 it has been assumed that node X_1 is randomly selected from the Grid as a first node then the distance between X_1 and the rest of the nodes in the Grid is calculated, where the closest node to X_1 is found and specified. Once, the closest node is specified it is then considered the Best Matching Unit (BMU). Then we start updating the values of all the nodes in the Grid according to the input which is X_i e.g., the BMU found. Note, that a certain range is edited by a certain percentage. This update happens with different values but at a close range to the BMU (e.g., Sensor Node) that is initially found. The main goal of the SOFM is that the BMU value will reach the same absolute value as X_1. This process is iterated from one node to another (X_1, X_2…X_i), where another node in the Grid in this case X_2 is chosen for the same calculation of applying the Euclidean distance and comparing and reapplying the same process [Kohonen, 2013].

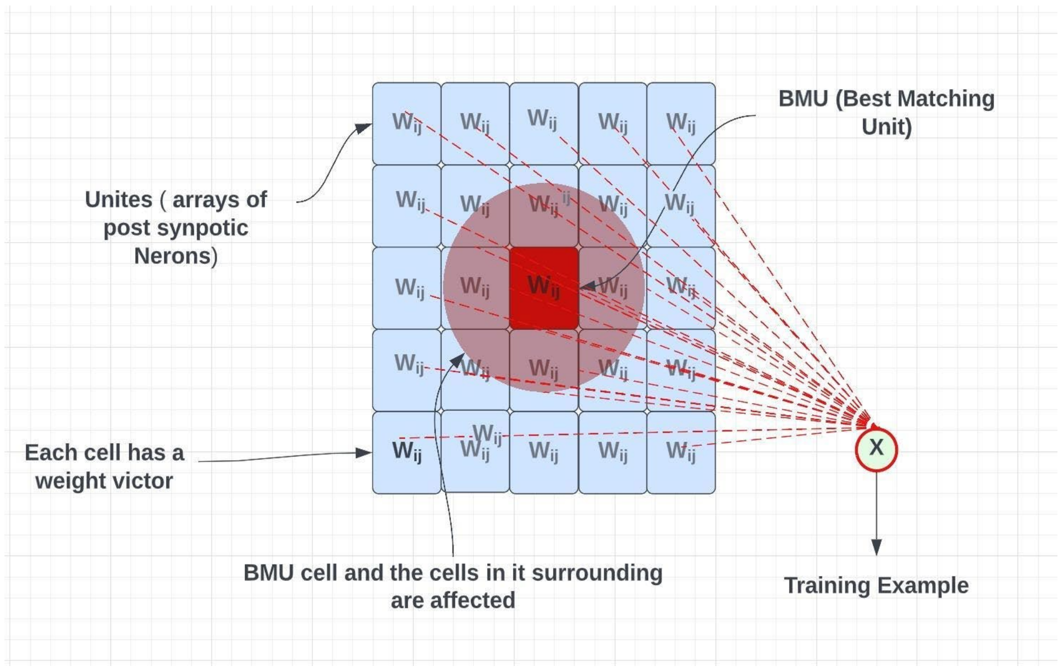

BMU (Best Matching Unit)
Unites (arrays of post synpotic Nerons)
Each cell has a weight victor
BMU cell and the cells in it surrounding are affected
Training Example
X
Wij

Figure 6.2 Updating the Best Matching Unit and its Neighbours.

In a typical SOFM environment, the optimisation algorithm works towards decreasing the problem-to-solution domain from multiple dimensions (e.g., 32) to limited dimension (e.g., 3) which makes it very easy to handle. It allows us to handle a large amount of data into manageable datasets without losing its functional properties with respect to the optimized solution. The main objective of this chapter is to further optimize the Bat node scheduling algorithms by using the SOFM optimization technique, which is based on update functions, unlike other algorithms like ANN which are based on activation functions.

The work in this chapter is based on enhancing the Bat node scheduling on unsupervised learning, therefore the HMM was not included in the optimisation efforts in this chapter because HMM and Principal component analysis are considered supervised learning which lies outside the focus of this chapter.

6.3 Proposed SOFM Node Scheduling Algorithm

The state space is represented/visualised spatially where a cluster type of WSN is assumed. The proposed SOFM node scheduling algorithm considers five features to provide the schedule solution: (1) the position of cluster heads with respect to the Sink, (2) the distance between the parent to child nodes, (3) the energy levels, (4) the Transmission link (Tx), and (5) the Receive link (Rx).
As SOM utilises the above-mentioned features to model the scheduling solution, it is known as SOFM. SOFM then can be used to find the winner node consisting of the best of the available features amongst all other available nodes. These features network is fed into the scheduling algorithm to provide the best scheduling process. The scheduling algorithm is based on the energy levels of the sensor node as it considers the distances between the Cluster Head and sink, parent and child node distances, Tx, and Rx. This energy feature function literally controls the other features on the formation of SOFM which is used in the optimizing process of the features in WSN.

In the SOFM model, the nodes are classified on the basis of 5 features, these features are in turn used to create the SOFM node schedule for the WSN. The SOFM partitions the network in the form of 5-D space layers. In order to illustrate the principle of the 5-D space layers, let's consider three layers starting from 1 to 3 where (layer 1) is presented for the information dissemination, e.g., packet exchanging, etc. As the WSN progresses in time, after SOFM iterations, some sensor nodes will experience a change in the level of energy i.e., consume energy, those sensor nodes are then replaced with other nearby available nodes that have the same features from the next layers e.g., updated from (layer 2) and consequently (layer 3). After some time, the layer under consideration will exhaust the solutions, now the onus is on the immediate next layer to take on the responsibility of the running node present.

6.4 Problem Formulation

To explore the search space and find the optimal solution for the MOO problem, especially, in the context of the WSN safety-critical systems. a multidisciplinary approach was used (i.e., that combines the optimization of three objectives: coverage, connectivity, and network lifetime) utilising the SOFM model spatial feature to formulate and solve the MOO problem. Therefore, this work proposes a novel node scheduling algorithm Using the SOFM node scheduling algorithm. In the SOFM node scheduling algorithm, nodes are grouped according to their similar features. For example, in the WSN, nodes share similar features such as energy levels, Transmission (T_x), Receiving (R_x), and distance (D) from the sink node or distance between the parent to child nodes, and the fifth feature is the sleep/ awake scheduling. The main purpose is to reduce high-dimensional data to a map.

Henceforth, assuming a data set of size (m, n) where m is the number of training examples e.g., the number of sensors in the WSN. While n is the number of features, a matrix of (100, 2) initialises weights (n, C) where C is the number of clusters. Iterating over the input data for each training example, it updates the weight vector with the shortest distance (Euclidean distance) from the training example. The updated rule of weight is represented in Equation 3. The general structure of the SOFM node scheduling algorithm is represented from [Kohonen, 2013] where we start the formulation by calculating the Euclidean distance (e.g., the shortest distance between two vectors) as in Equation 6.1 :

$$D = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{0.5} \qquad (6.1)$$

Where D is the Euclidean distance, x and y are the cartesian coordinates of the nodes in the field. where (x_1, y_1) and (x_2, y_2) are the input coordinates of the randomly selected vector /neuron. The mc is the Codebook (feature set) vector matching a given input vector X. as described in Equation 6.2:

$$x - mc = min\{x - mi\} \qquad (6.2)$$

The mi is the average spatial x-coordinate of a neuron. In this work, we formally defined the mc as the neuron (sensors).

The SOFM units receive updates, as well as their neighbours where the unit of the highest match is achieved. Within such a process, the best unit becomes updated closer towards the sample vector in the space of input value. In Equation 6.3

$$w_{i.j} = w_{i.j}[old] + alpha\,[t] * [x_{i.k} - w_{i.j}[old]] \quad (6.3)$$

where $alpha$ is a learning rate at time t, j denotes the winning vector, i denotes the i^{th}feature of the training example and k denotes the k^{th}training example from the input data. After training the SOFM network, the trained $w_{i.j}$ weights are used for clustering new examples. A new example falls in the cluster of winning vectors. The first step is to find the distance between two sensor nodes. For example, if the distance between, say, node *A* and node *B* is found then they will be compared, at that point the distance between sensor *B* and sensor *C* can be found as well. The calculations for clustering new examples are calculated by Equation 6.4:

$$d\_E\,[x, y] = \|x - y\| \quad (6.4)$$

Where $\| \ \|$ represent the Euclidean distance can be found in terms of the Euclidean norm of the difference between two vectors (sensor nodes). Please note that *A* and *B* represent two sensor nodes in the system as shown in Equation 6.5:

$$A[x_1, y_1], B[x_2, y_2] \quad (6.5)$$

Finally, the Best Matching Unit (BMU) is updated with respect to the next immediate neighbour, usually noted as m_c which is represented by Equation 6.6

$$\big\|x - m_c\big\| = min_i\{\big\|x - m_i\big\|\} \quad (6.6)$$

Equation 6.6 represents the normalization of the input vector X and the output vector, where the value of the output vector is equivalent to the minimum Euclidean distance between normalized input vector and the path weight vector.

The purpose of Equation 6.6 is to identify the nodes which can be CH nodes i.e., the nodes have better position and energy levels. For example, if 3 sensors are considered, *A*, *B*, *C* logically there will be a path from *A* to *B*, and a path from B to C, hence there are two paths which are two distances noted by *D0* and *D1*. If *D0* > *D1*, then the *B* sensor is not CH, otherwise then it becomes a CH. Operating Equation 6.6 function might find more CHs than required, hence another function is needed which will select the best 5 CHs among them. The second function updates the weights and assigns each weight to the sensor node. This function will collect the highest weights assigned to the sensor nodes. Hence, the Pseudocode of the SOFM node scheduling algorithms is presented in Algorithm 6.1.

Algorithm 6.1 SOFM node scheduling algorithms

```
Input:   1. Input data = "sensor_data.txt" (five features of
            sensors) which is represented in a tabular form in csv
            format
         2. Number of Sensor Nodes = 10 (100)
         3. Number of epochs = 100 (1000)
         4. learning_rate = 0.1 (0.1)
Output:  SOFM Optimized 5 features for creating efficient
         network for WSN.
1.  Define function to train SOFM (box)
2.  Initialize SOFM weights with random numbers between 0 to 1
3.  for i = 1 to num_epochs
4.      Loop through input data
5.      Calculate the best matching unit (BMU) for the input
        data
6.      Update the weights of the BMU and its neighbors
7.  End for
8.  for i = 1 to (number of nodes):
9.      calculate_distance (updated weights of BMU, Input
        data)
10.     if distance < min_distance:
11.         min_distance = distance
12.         BMU Index = i
13.     End if
14. End for
15.   return BMU Index
16. Define function to optimize the network in terms of energy and
    communication
17. Use the trained SOFM to extract features from sensor nodes
18. Use the extracted features to optimize the network
19. for i = 1 to (number of nodes):
20.     calculate the distance to each SOFM node
21. End for
22. Assign the node with the minimum distance as the feature for
    the sensor node
23. Obtain the SOFM Optimized 5 features for creating efficient
    network for WSN.
```

The operational setup of the SOFM algorithm includes the five features mentioned earlier, each feature is associated with a certain weight value which needs to be updated in every iteration based on those features. Here the Euclidean distance is calculated, and the best matching is identified where the clustering starts to form a group as a potential solution. Each group would have its own sleep scheduling which is based on a value obtained from the features. All these processes are calculated by the sink node during the design time and implemented in the run time. Figure 6.3 illustrates the output of the SOFM node scheduling algorithm for creating an efficient network for the WSN.

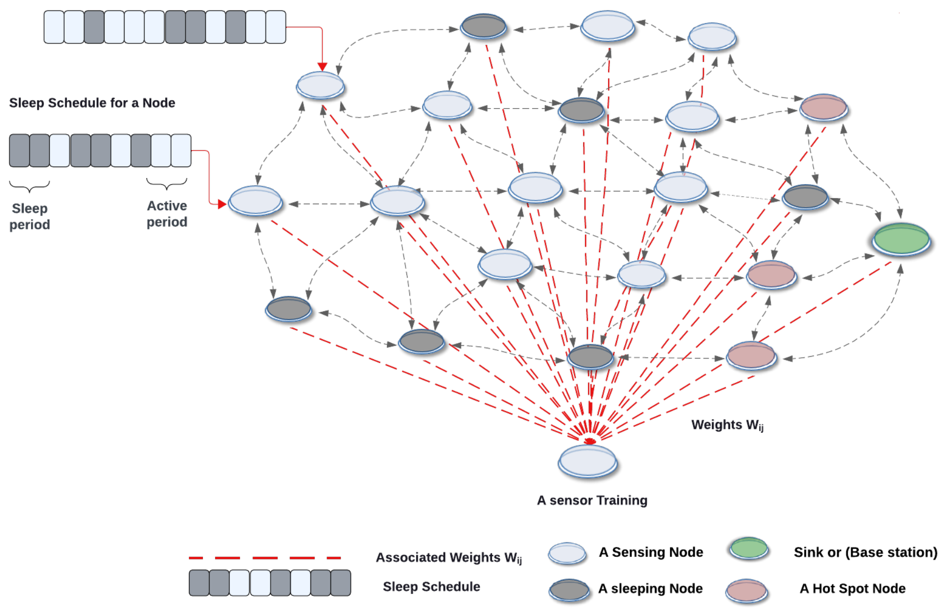


Figure 6.3 The Visualization of the WSN using SOFM Node Scheduling Algorithm.

These features are fed into the SOFM model to provide the best node scheduling process where the SOFM node scheduling algorithm can be used to find the winner node consisting of the best of the available features amongst all other available nodes. The mechanism of the SOFM node scheduling algorithm is based on the energy levels of the sensor node and the distances between the CH and sink, parent and child node distances, T_x, R_x, and the node sleep/ awake scheduling. Thus, the SOFM node scheduling algorithm partitions the network in the form of five-dimensional space layers.

In order to illustrate the principle of the five-dimensional space layers, let's consider a simple example of three layers starting from one to three where (layer one) is presented for the information dissemination, e.g., packet exchanging, etc. As the WSN progresses in time, after SOFM node scheduling algorithm iterations, some sensor nodes will experience a change in the level of energy i.e., consume energy, those sensor nodes are then replaced with other nearby available nodes that have the same features from the next layers e.g., updated from (layer two) and consequently (layer three). After some time, the layer under consideration will exhaust the solutions, now the onus is on the immediate next layer to take on the responsibility of the running node present.

6.5 Experimentations Set-Up

The simulation experimentations were created in the MATLAB environment, where the following parameters were used as illustrated in Figure 6.4. The simulation consists of 100 homogenous nodes i.e., having the same capabilities (sensing, communicating, power battery and processing power). The WSN monitored area in simulations is 100 m^2.

1. All nodes are connected to one sink node.
2. Nodes are randomly deployed.
3. The number of iterations is fixed to 200 epochs.
4. Number of simulation iterations is limited by convergence of the SOFM network, when the wait time reaches a stable condition.

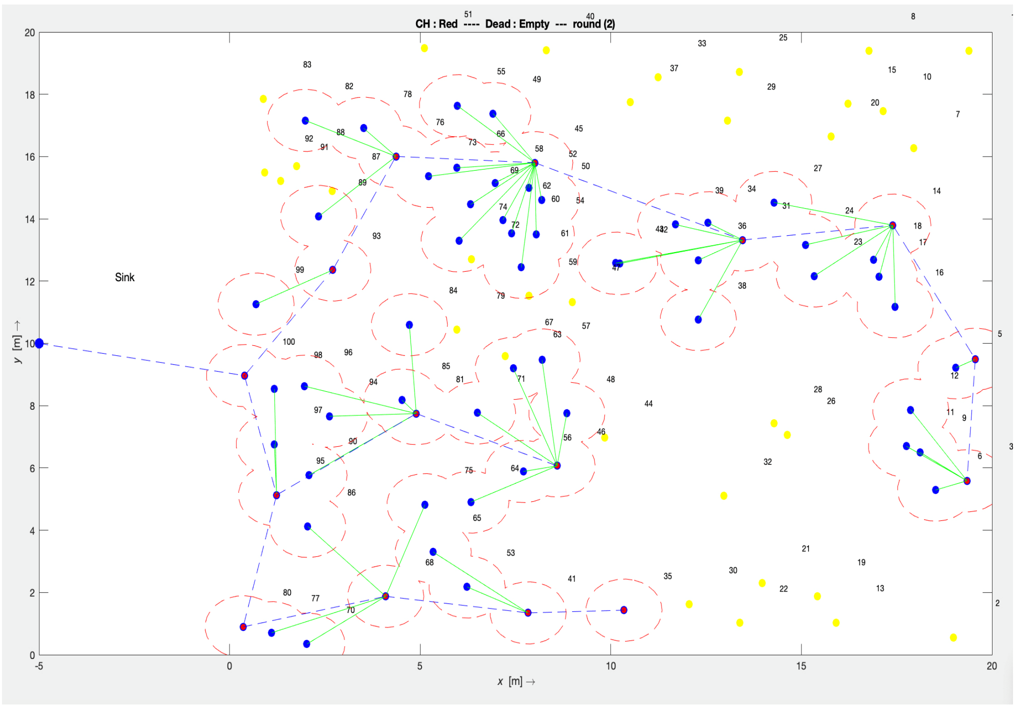


Figure 6.4 Simulation Set-up of the SOFM Node Scheduling Algorithm

In the experimentations, without the loss of generality (all the assumptions considered in this literature are valid.) the SOFM algorithm was compared to the Bat node scheduling algorithm. In figure 6.4, the red node is the Cluster Head, the blue node is the Child node, and the Yellow node is the Dead Node. Comparisons were not included between SOFM and other algorithms (like RCS and HMM) because they were demonstrated in previous chapters showing how the Bat node scheduling algorithm outperforms the RCS and HMM. Therefore, if improvements are shown from the SOFM over the Bat node scheduling algorithm, it can be deduced SOFM also outperforms RCS and HMM algorithms.

The simulation was individually tested 30 times, where one-way ANOVA analysis was implemented on both the SOFM and the BAT node scheduling algorithms to identify the p-value for the statistical analysis, the statistical results are shown in Table 6.1.

Table 6.1 Statistical Analysis

Metrics	Bat Algorithm	SOM Algorithm
Number of Hops	26	28
Connectivity	0.99	0.99
Coverage	0.77	0.99
Lifetime	196	126
ANOVA 1	4.12002e-188	6.04755e-89

6.5.1 All Used Energy

The all used energy metric refers to the energy consumed by the nodes throughout the simulation time until all nodes fully deplete/consume their energy. In Figure 6.5, the X-axis represents time in round units, and the Y-axis is the energy used in Joule.

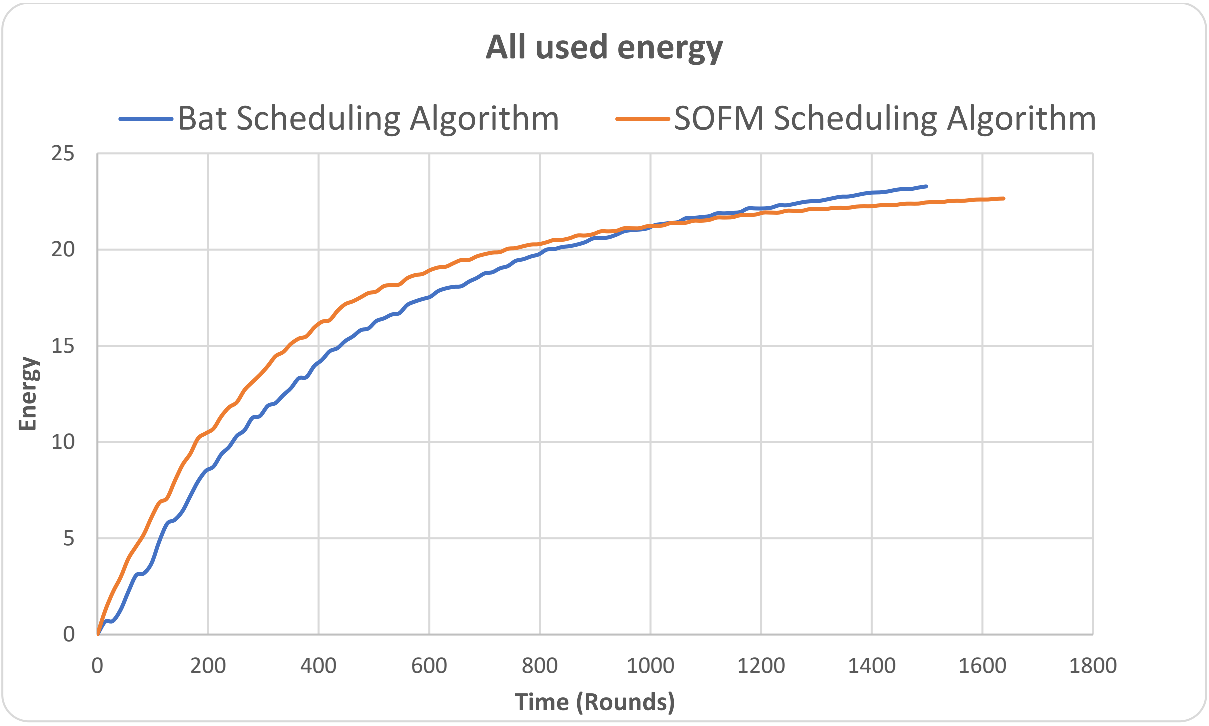


Figure 6.5 All Used Energy

For this metric, the SOFM algorithm was able to collectively consume less energy by the WSN as a whole and last longer up to the 1638th rounds, while the Bat node scheduling algorithm lasted up until the 1498th rounds only. This observation shows the SOFM algorithm consumed less amount of jealous energy hence, extended the network's lifetime more than the Bat node scheduling algorithm. Albeit these results, the Bat node scheduling algorithm did consume less energy during the first half of the simulation. As, for instance, can be noticed in the 42nd round with respect to the Bat node scheduling algorithm, the amount of energy used is 1.28 joules in contrast to the amount of energy used in SOFM is 2.97 joules. Figure 6.5, observation shows that at the 1st round to the 1000 rounds that the SOFM algorithm consumed more energy as compared to Bat node scheduling, but still it was able to compensate with the Bat node scheduling algorithm performance in consuming energy during the last performance for 1000 to finish. This extraordinary performance is entirely due to SOFM algorithm simplicity and effectiveness in finding the global minima in terms of energy of the network. The setup mechanism of the SOFM algorithm was based on features mainly the residual energy feature, thereby, this visualises the performance of the WSN in terms of the best topology that should achieve the optimal QoS.

6.5.2 Lifetime of Sensor Nodes

The lifetime of sensor nodes metric represents the number of live sensor nodes in the network during the simulation rounds. In this metric, the Y-axis represents the number of alive sensor nodes, while the X-axis represents times in rounds as shown in Figure 6.6.

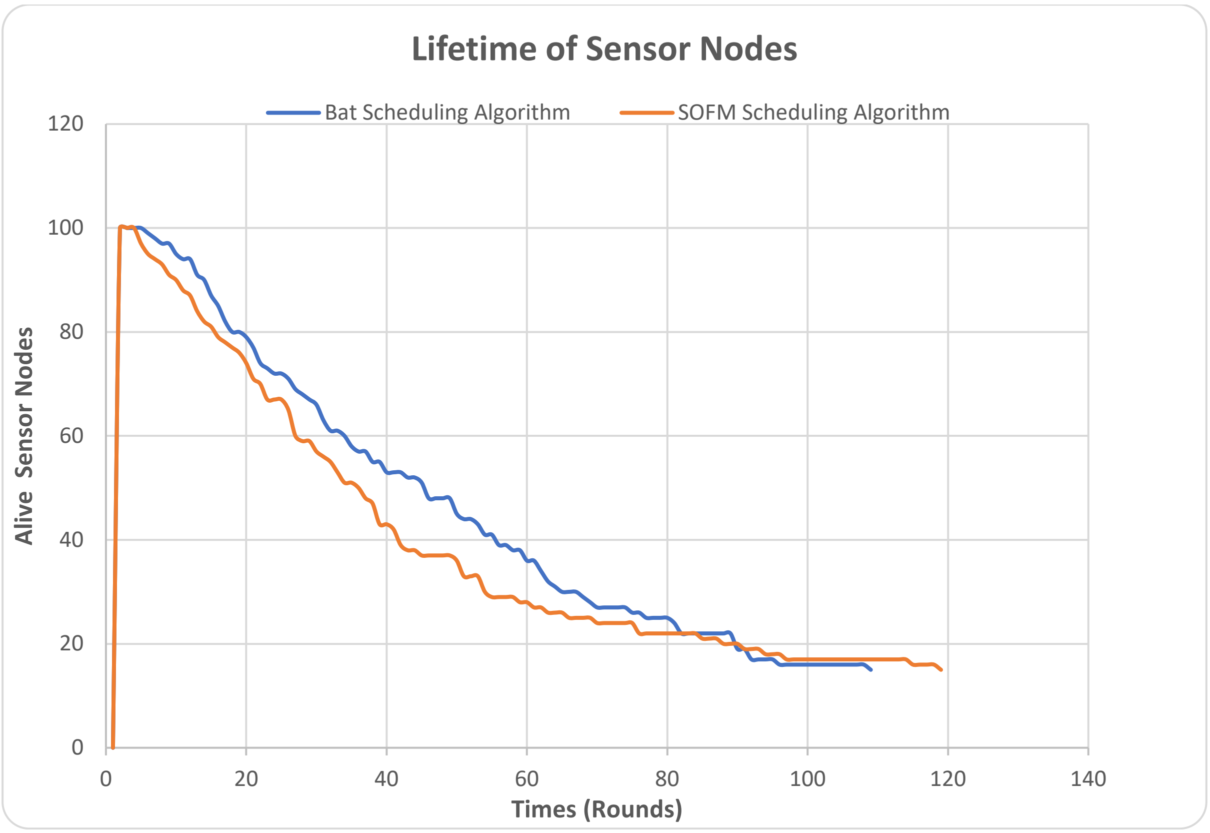


Figure 6.6 Lifetime Sensor Nodes

Initially both algorithms in the WSN started their operation with 100 nodes. It can be noted from the Figure that SOFM falls below Bat node scheduling from round 0 up to 85th rounds, this is not a disadvantage by the SOFM algorithm because the higher the number of the live sensors the higher the energy consumption which is an undesired result. It can be noticed that Bat node scheduling algorithm fully depletes its energy at the 108th round where it is recorded to have only 15 nodes alive, where the simulation threshold is set to finish when the number of nodes reaches 15. In contrast, the SOFM algorithm has maintained the live sensors for a longer period until the 118 round which is an advantage by 10 rounds over the Bat node scheduling algorithm. Figure 6.6 shows similar pattern as the SOFM algorithm the orange curves tends to compensate with the Bat node scheduling algorithm blue curve performance at the 83th rounds till 97th rounds then it out performed from the 96th rounds to 119th rounds. This is due the features consideration mainly the residual energy in visualizing the self-organizing feature map of the WSN.

6.5.3 Efficiency

The X-axis is the number of rounds, and the Y-axis is the number of packets generated in the network which reflect the congestion of the packets in the network as shown in Figure 6.7, the less packets generated the less congestion and the more packet generate is the more the congestion which will results to eventually impact the nodes to deplete energy. The main point of interest is the number of packets sent and received in the WSN.

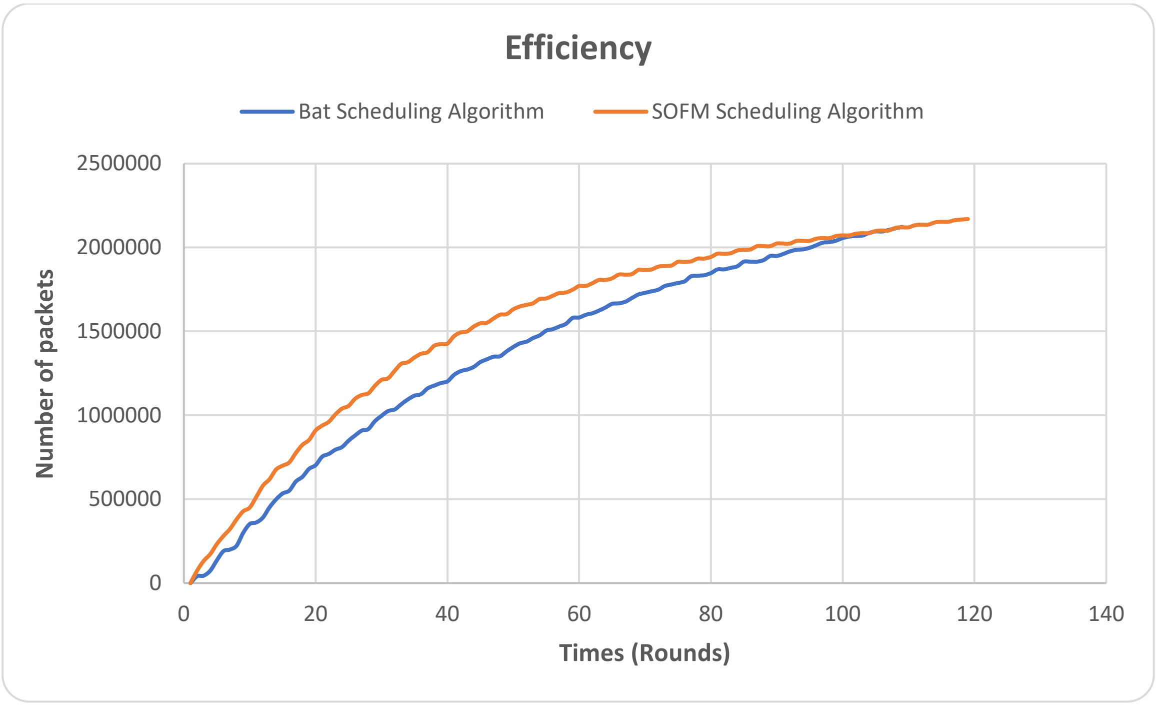


Figure 6.7 Efficiency.

After 110th rounds, it can be noticed SOFM algorithm has recorded a higher Efficiency with extended Efficiency-lifetime too where the Bat node scheduling algorithm recorded a network lifetime at the 110th round (died). The SOFM algorithm has the larger number of packets sent/received in the network without any jitters which indicates better connectivity and lifetime of the network with greater expense of useful energy which is available in SOFM than in Bat node scheduling.

6.5.4 Connectivity

The Y-axis is the performance of connectivity which is measured by number as a value fraction of nodes connected in the network. For example, in the simulation, if there are 100 nodes in the network as shown in Figure 6.8, if the connectivity is 0.98, then the total connected nodes are 100*0.98=98. The simulation experiment is comprised of 120 sensor nodes. The X-axis is the number of rounds.

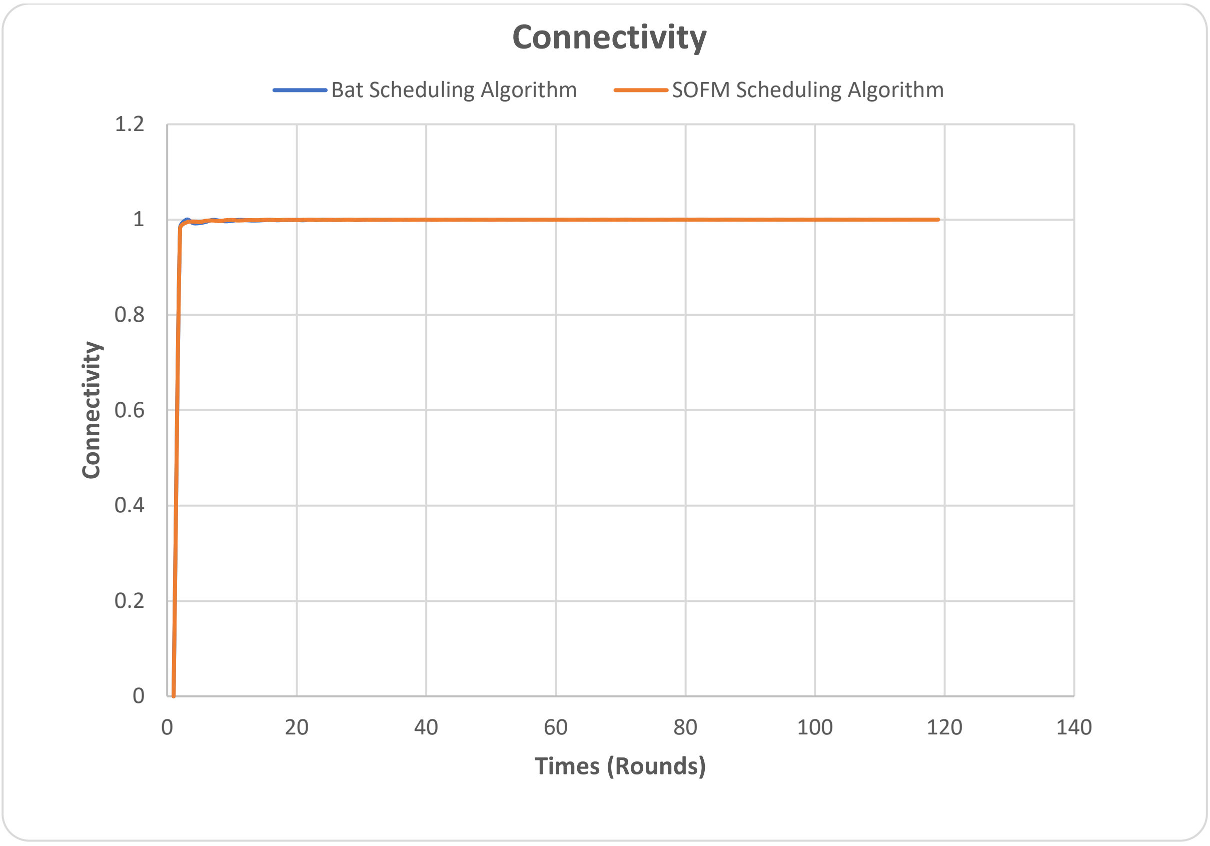


Figure 6.8 Connectivity.

It can be noticed that both algorithms performed similarly well during the simulation period with the exception of the SOFM's extended network lifetime as can notice the Bat node scheduling algorithm stopped/died at the 108th round, where SOFM continues its life cycle until the 118th round. This attributed by the simplistic nature of the SOFM preserving more energy to be used at later stages in the network.

6.5.5 Coverage

In figure 6.9, the Y-axis is the performance of coverage which is measured by number as a value fraction of nodes connected in the network. In the simulation, the coverage is represented by the fraction of area covered by the network. If the area of the test field is 100m^2, the coverage of 0.8 would mean 0.8*100 = 80m^2 has been covered. The X-axis is the number of simulation rounds.

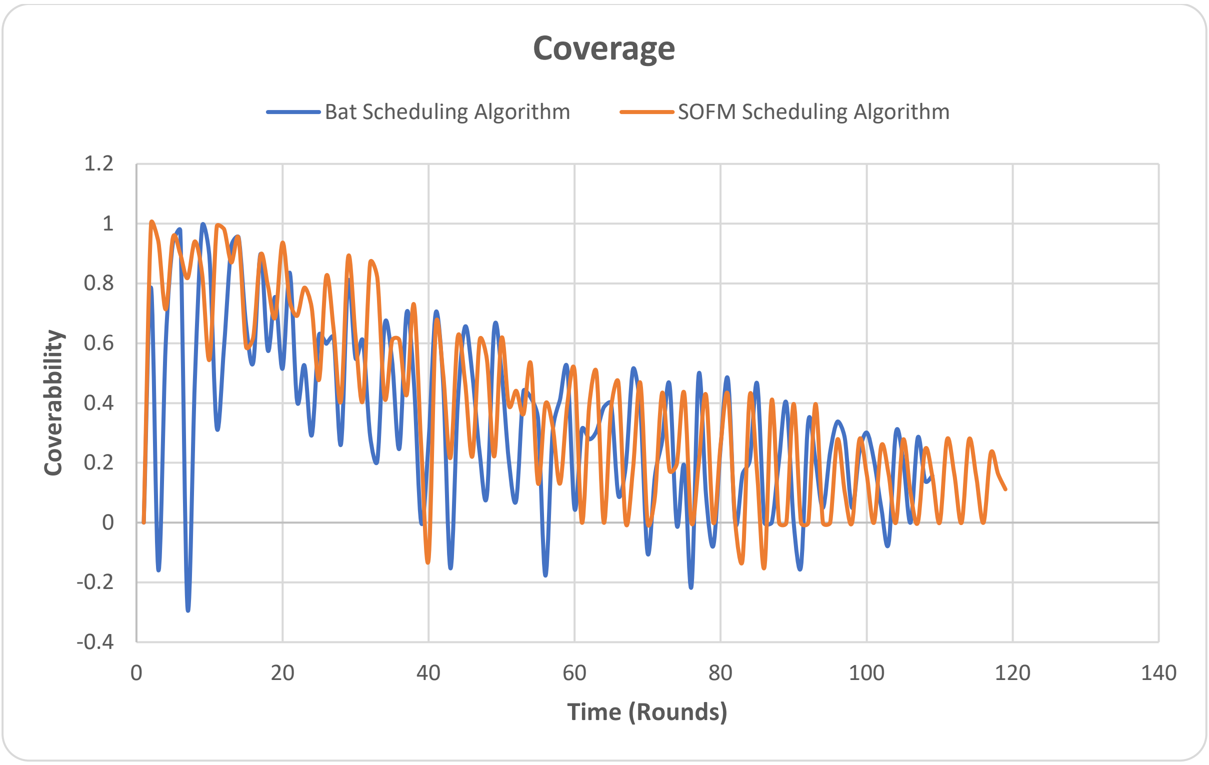


Figure 6.9 Coverage.

The overall coverage as seen in Figure 6.9 fluctuates as time progresses, as the nodes' battery power falls the coverage decreases. Similarly, to previous metrics, after the 108th round the Bat node scheduling algorithm dies off and hence no coverage is seen but the SOFM carries the coverage up to the 118th round due to the availability of the SOFM algorithm preserving more energy to be used at later stages in the network.

The sharp negative fluctuations in the network are generated mainly by two causes.

1. The energy holes
2. The delay in communication between the BS and the concerned sensor node.

Meanwhile, the positive fluctuations are provided by both SOFM and Bat node scheduling algorithms from the 60th rounds to the 108th round. It can be clearly noticed from the blue curve in Figure 6.9 that the Bat node scheduling algorithm provides the same level of coverage which is 0.4% as the SOFM algorithm (orange curve). However, the orange curve extends its stabilization to outperform the blue curve, and this is due to the stochastic mechanism of the SOFM algorithm.

6.5.6 Evaluation and Interpretation

This chapter presented a new node scheduling algorithm based on the SOFM model to provide a better optimal solution for WSNs. The achieved results showed an improvement in service availability, reliability, and network lifetime in comparison to the Bat node scheduling algorithm. The spatial feature of the SOFM enabled a better representation of the WSN configuration, hence, providing a better performance solution with respect to all three objectives of the MOO problem. In particular, the self-organizing capacity in the SOFM algorithm outperforms the Bat node scheduling algorithm through its simplistic method of dimension reduction and classification. The SOFM algorithm has outperformed the Bat node scheduling algorithm in all metrics, whereas in some metrics both have scored similar values, however, the SOFM algorithm always provides an extended network lifetime over the Bat node scheduling algorithm. Hence, the output results are shown to be better than the previously proposed algorithms e.g., RCS, HMM and Bat node scheduling algorithms by far in this thesis.

The cost of implementing the Bat node scheduling algorithm is higher as compared to the SOFM algorithm [chapter 6]. In the Bat node scheduling algorithm, there was a complexity level concerning operational design, fitness and objective function settings coupled with the Pareto sorting algorithm to select the best possible solution. The SOFM is a method which is based on a neural network structure that constitutes dimensionality reduction. In general, dimensionality reduction is used to represent high-dimensional datasets as a two-dimensional dataset with an abstract pattern that contains the best solutions. After the convergence is achieved the weights associated with the features are stable, this provides a topology-preserving mapping from the high dimensional space i.e., lossless data compression occurs.

The SOFM node scheduling algorithm is designed for spatial-related spaces rather than in time-dependent spaces, so the data representations may be abstracted due to the migration from spatial to temporal spaces. In the scheduling scheme, the time length of 30 minutes will be represented approximately by the SOFM with pointers to the time difference. The next chapter addresses this limitation by introducing the LSTM networks whose approach is entirely on a time-dependent framework.

Chapter 7 A New Scheduling Algorithm for Improved Performance of Multi-Objective Safety-Critical WSN Using LSTM

7.1 Introduction

This chapter addresses the MOO problem by exploring a different representation of the solution space, in which time series data prediction is utilised, using the LSTM model. The time series data prediction method refers to the process in which the occurrences of important future processes is predicted [Widiasari et al., 2018]. One approach to address the MOO problem is to find time series data of good coverage patterns, through the performance analysis of the WSN. Then, replicate this time series of data coverage patterns to obtain a better performance throughout the entire lifetime of the WSN. This shall achieve a state of improved coverage stable connectivity and network lifetime. Many predictive analysis models exploit time series data to improve the system performance. For example, nonlinear prediction NLPs, ANN, and HMM are all data prediction analysis models to improve accuracy. However, the difference between the formers and LSTM is that LSTM can gather, process, and predict data sequences without forgetting unimportant information useful for time series prediction. Hence, a based scheduling algorithm is proposed to analyse and provide a suitable, energy-efficient scheduling solution.

Several node scheduling attempts in the existing literature show that there is still room for improvement. In particular, previous analyses [chapter 6] have yielded patterns in our coverage data, LSTM is a Recurrent Neural Network (RNN) based model which remembers the short useful patterns to predict the next sequential time series data [Mohanty et al., 2020]. Since LSTM can balance the temporal parameters in the WSN and provide the best possible scheduling sequence for a given system, in this work an LSTM-based node scheduling algorithm is proposed to provide an energy-efficient scheduling solution where the system output here includes the following parameters: (1) Less Energy consumption, (2) Better Connectivity, (3) Best coverage and (4) best scheduling solution. The motivation behind choosing LSTM as one of the proposed solutions to solve the issue of the MOO in WSN is the following key features of the LSTM model [Cheng et al., 2019].

- First, the LSTM model is well-suited for capturing temporal dependencies and patterns in sequential data, known as Sequence Modelling. In WSNs, where sensor readings are often collected over time, the LSTM model can help WSN models predict patterns or events.
- Second, LSTM can automatically learn relevant features from raw sensor data, reducing the need for manual feature engineering, this is known as Feature Extraction.
- Third, LSTM can capture complex relationships and non-linearities in data, which can be valuable for detecting anomalies, predicting events, or performing other tasks in WSNs, this is known as non-Linearity.
- Fourth, LSTM can handle variable-length sequences, which is important when dealing with irregularly sampled sensor data (Flexibility).

Despite the many advantages the LSTM offers, several challenges be taken into consideration when utilising the LSTM model in WSN. The following is the list of challenges that revolve around the LSTM implementation [Salmi and Oughdir, 2022]:

- LSTM can be computationally intensive, which may pose challenges in resource-constrained sensor nodes with limited processing power, memory, and energy.
- LSTM might be challenging in collecting and storing data for training resource-constrained environments. In WSN, data transmission is limited to the power battery of the sensor node, thus it could consume significant energy which makes it challenging to collect and store data for training WSN.
- LSTM is prone to overfitting when training data is limited. Proper regularization techniques and careful model selection are important to mitigate overfitting.
- LSTM may struggle with noisy or missing data sets obtained from real-life sensor networks, as WSNs may have lossy wireless links. Therefore, the reading from the sensor nodes can be unreliable. Hence, pre-processing and data cleaning is crucial before passing it into the LSTM model to mitigate the data quality issue.
- LSTM is often considered a "black box" model, making it challenging to interpret the reasoning behind its predictions, which may be important in some applications (known as Interpretability).

In the WSN context, assuming a sensor network that can run a maximum of 2000 rounds after each round, say round 1, the LSTM models will predict the best possible configuration for the current rounds. This configuration is the best possible sequence of nodes predicted from the precious in here 2nd round. Consequently, the 3rd round would be the best prediction of its predecessor which is the 2nd round. This process of prediction will continue until a particular threshold of energy is achieved.

7.2 Long Short-Term Memory

As stated earlier, the LSTM model is based on the RNN training model of the error gradient to vanish or explode out (error-free). For example, if there is an error gradient value of 0.1 and another error gradient value of 0.1, the multiplication of both errors would be 0.01 which means the error rate is increasing. In this situation, there will be inaccurate predictions in both cases due to the underflow or overflow of the error gradient. Hence to overcome this situation we must introduce a cell which will be responsible for memory in the short term. The predictions must have errors where the smaller the errors the more accurate the predictions, hence, the network is therefore learning to predict. Unlike prediction, reproduction memorizes the object with zero errors and when the error gradient tends to zero, reproduction here is undesirable.

The aim is to ensure as accurate a prediction as possible with the least errors while maintaining a second objective for increasing the prediction range by introducing a FORGET GATE parameter. A simple example of these two objectives is illustrated in a text prediction task where, say, we want to predict a complete sentence of “I want to go to London” - by checking the error gradient, our prediction might have a gradient error of 0.1 which may produce an output sentence “I am going to Dodnon” we can notice Dondon instead of London. The main aim here is to work on our gradient error so we can have as much as possible of a near-optimal solution. LSTM can remember the sequence of events that occurred longer than the case of reproduction of this output. The short sequences are stored in short-term memory whereas the long sequences which are a combination of short sequences are stored in long-term memory to form useful patterns for prediction purposes.

To continue with the same example for the FORGET GATE parameter, this gate will be triggered after a certain pattern is complete (e.g., the example of the text above “I want to go to London”) and then it starts a new pattern. If we want our text prediction task to predict a longer sentence “I want to go to London and catch a connecting flight to Dublin to meet a friend". Here, we have two parts of the sentence, whereby introducing the FORGET GATE our network can control what to forget and what to remember. In our case here, it will forget the first part which is "I want to go to London" and it will remember the second part which is "and catch a connecting flight to Dublin to meet a friend", hence our prediction task will be more efficient. The process of the first sentence is finished and completed where LSTM is used to predict the second sequence of the sentence which needs to be processed and predicted too. LSTM classification can be found in many real-life use cases, for instance, in text predictions, voice recognition and stock market predictions, as well as computing WSNs.

7.3 Problem Formulation

The proposed LSTM node scheduling algorithm manages the scheduling process and the behaviour of the sensor nodes in WSN concerning performance throughout discrete-time sequences which is time-bound multiplexing. For example, dividing the timeline (T) between the number of sensors (S) in the WSN as $T_1=S_1$ and $T_2=S_2$ etc. The LSTM network will recognize patterns that are short and efficient in terms of energy. Considering the following sensor cycle invocation at an energy level, say 10 Joules and sensors range of $S = \{(S_1, S_2, S_3, S_4)\}$ where these are considered efficient patterns. The next best pattern will be obtained from the prediction of the LSTM network as trained from the WSN. The output of the LSTM network will be energy function per range which will be used as a projection to the next cycles where it can be suitably used to provide the best cycle for dependability and efficiency throughout the WSN lifecycle. In our previous experiments, we have managed to analyse and find that coverage data include "good" and "bad" results in some zones the sensor nodes performed better than other sensors in different zones. Hence, the classical algorithm RCS, IHMM, BAT and SOFM algorithms cannot capture the best coverage patterns. The analysis of the coverage metrics outputs random fluctuation behaviour, see the coverage graphs in Figure 4.7, Figure 5.8, and Figure 6.9 in the previous chapters.

To reduce the fluctuation, the LSTM algorithm is used to predict good and meaningful patterns as far as practicable. Therefore, a new LSTM node scheduling algorithm is proposed to capture good and meaningful coverage patterns and replicate them for the entire cycle of the WSN. Hence, providing a good improvement in the availability and reliability of the WSNs network services.

LSTM node scheduling algorithm introduces, the new, the following parameters: an input x_t , where subscriptions the current Energy level, a where h_{t-1} is the previously consumed Energy level and c_{t-1}, where is a dependent parameter on energy level at a time $t - 1$, an o_t is the output of the LSTM algorithm for this time step. The LSTM also generates the c(t) and h(t) for the consumption of the next time step t as illustrated by Figure 7.1.

$$f_t = \sigma_g(W_f \times x_t + U_f \times h_{t-1} + b_f) \quad (7.1)$$

Where $f_t$is the forget gate

$$i_t = \sigma_g(W_i \times x_t + U_i \times h_{t-1} + b_i) \quad (7.2)$$

Where $i_t$is the input gate

$$o_t = \sigma_g(W_o \times x_t + U_o \times h_{t-1} + b_o) \quad (7.3)$$

Where o_t is the output gate

$$c_t' = \sigma_c(W_c \times x_t + U_c \times h_{t-1} + b_c) \quad (7.4)$$

$$c_t = f_t . c_{t-1} + i_t . c_t' \quad (7.5)$$

Where $c_t$is the cell gate

$$h_t = o_t . \sigma_c(c_t) \quad (7.6)$$

Where $h_t$is the hidden state

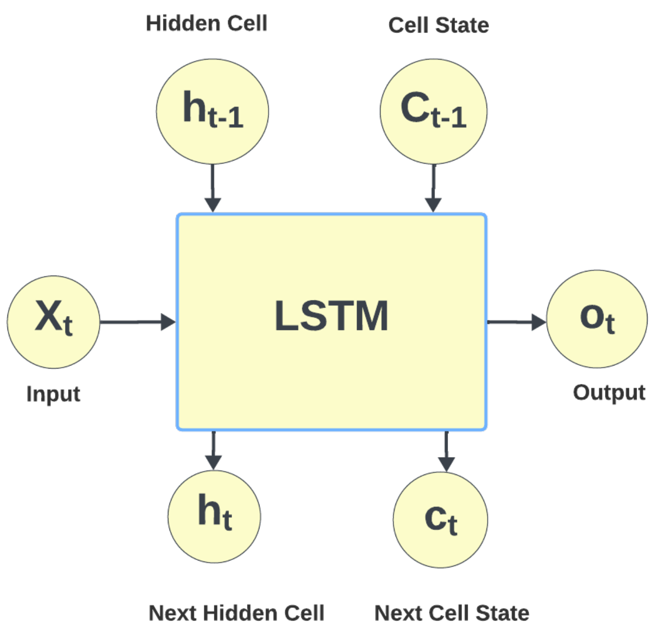


Figure.7.1 Utilisation of LSTM model in WSN scheduling approach.

LSTM node scheduling algorithm uses the current state of the network e.g., each sensor node energy level at time t, to predict the real-time energy of the network. Hence, the solution is time dependent. LSTM is a memory model which is time-dependent thus the future step can be predicted based on the current step. Unlike the SOFM node scheduling algorithm which is a space model that is solely based on the spatial configuration of the network hence a feature-based solution is provided.

7.4 The proposed LSTM Node Scheduling Algorithm

The LSTM model operates independently and performs the training during the WSN rung time. As the WSN start its operation, a dataset is generated, and the LSTM scheduling algorithm then starts its training process. The dataset is taken as inputs to the LSTM machine learning algorithm. The dataset consists of maximum connectivity and coverage values that correspond to subsequent energy values known as data points. These data points are fed into the LSTM algorithm to scour, match, and filter the same patterns of data points from the rounds of the WSN while it is on runtime.

For a better understanding of the mechanism of the LSTM algorithm, suppose we have the first 10 rounds generated from the WSN with rounds 4th, 5th, and 6th rounds showing interesting patterns of maximum coverage and connectivity, energy, and scheduling coordinates points. Subsequently, for the next 10 rounds, these patterns of data points are identified by using the LSTM learning method and released as the output of the LSTM algorithm. The LSTM algorithm will cease to run until the threshold energy level of the sensor node falls below the operating energy level of the node.

Finally, this output is stored as a logbook and is used successfully by the same WSN without using the LSTM Algorithm. Figure 7.2 explains the mechanism of the LSTM Node Scheduling Algorithm.

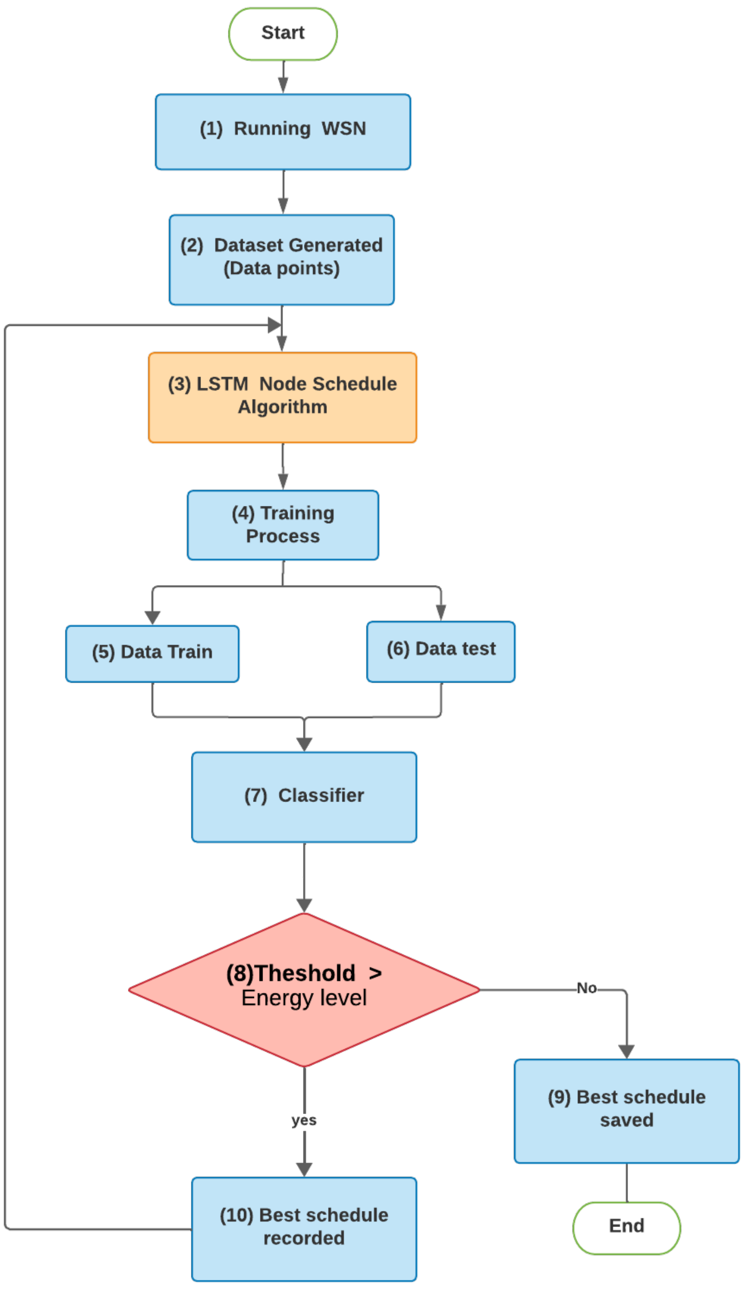

Start
(1) Running WSN
(2) Dataset Generated (Data points)
(3) LSTM Node Schedule Algorithm
(4) Training Process
(5) Data Train
(6) Data test
(7) Classifier
(8)Theshold > Energy level
No
(9) Best schedule saved
yes
(10) Best schedule recorded
End

Figure 7.2 The mechanism of the LSTM node scheduling algorithm.

The process of the LSTM Algorithm begins by running the network with a Node Scheduling Algorithm to collect the dataset required for the training process. Once an interesting sequence is identified, the algorithm detects and records it. Subsequently, this sequence of interest is replicated throughout the lifetime of the WSN. Thus, after illustrating the LSTM Node Scheduling Algorithm mechanism, Algorithm 7.1 presents step by step pseudocode of the LSTM mechanism.

Algorithm 7.1 The process of LSTM model implementation in WSN

- **Input**: Energy level values at t =t-1 and t=t, dependable parameters (coverage and connectivity)
- **Output**: Scheduling (Energy levels at t=t+1 (prediction), dependable parameters, (coverage and connectivity))

```
1: **Initialize** LSTM model
2: **Define** input shape based on sensor data dimensions
3: **Define** LSTM layers
4:        **Add** LSTM layer with specified number of units and input shape
5:        **Add** additional LSTM layers if necessary
6:                **Add** Dense layers
7:                **Add** dense layers with appropriate activation functions
8:                **Compile** the model
9: **Specify** loss function, optimizer, and metrics
10: **Train** the model
11: **Provide** training data (sensor data and corresponding energy consumption)
12: **Specify** the number of epochs and batch size
13: **Evaluate** the model
14: **Provide** test data to evaluate the model's performance
15:       **Use** the trained model for energy scheduling
16: **Collect** real-time sensor data
17: **Pre-process** the data to match the input shape of the LSTM model
18:       **Use** the model to predict the energy consumption for the given sensor data
19:       **Implement** energy scheduling algorithm
20: Determine the energy requirements and constraints of the wireless sensor network
```

21: **Use** the predicted energy consumption values to schedule energy usage and optimize resource allocation

22: **Consider** factors such as battery life, energy efficiency, network coverage, and data transmission requirements

23: **Repeat** the energy scheduling process periodically

24: **Collect** updated sensor data

25**: Pre-process** the data and feed it to the LSTM model for energy consumption prediction

26: Adjust the energy scheduling algorithm based on real-time sensor data and network conditions

7.5 Implementing LSTM in WSN

LSTM stems from RNN methodology with the objective of finding meaningful patterns with respect to the design's interest and detecting such patterns using training algorithms as illustrated in Figure 7.3 below.

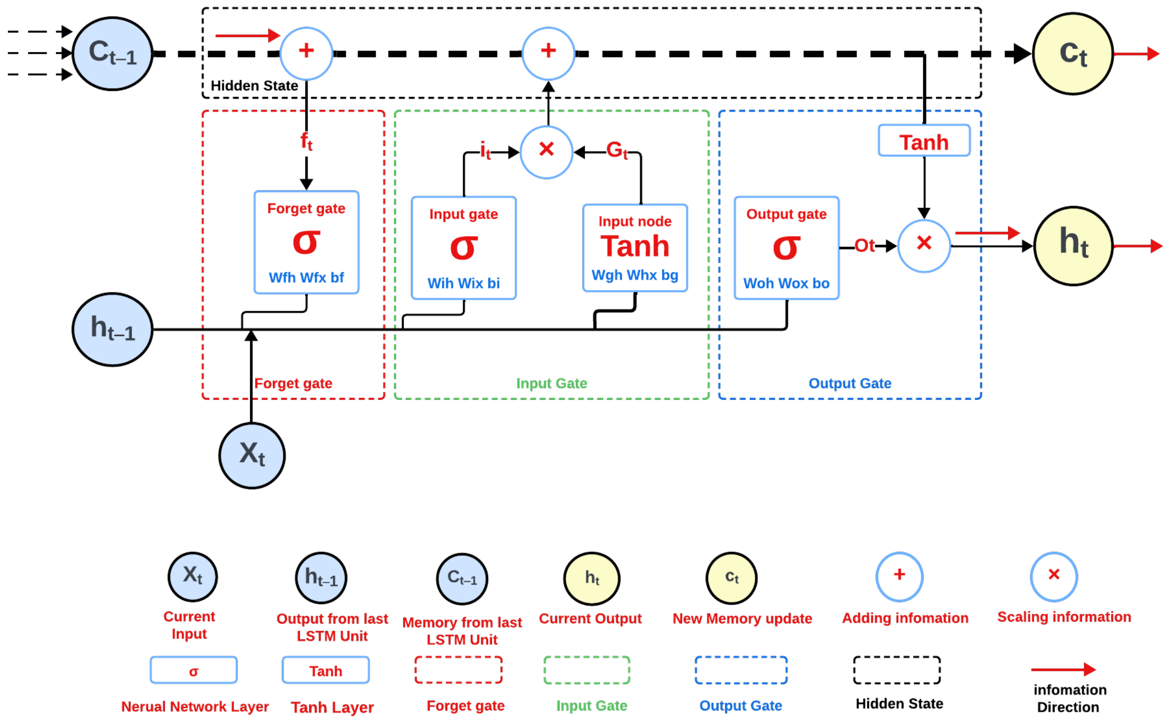


Figure 7.3 LSTM implementation in WSN

The LSTM architecture comprises several key components:

Sequential RNN Unit: The LSTM begins with an assembly of sequential recurrent neural network (RNN) units.

Cell state (LSTM): The core of the LSTM is the cell state. It serves as a memory unit that stores information from past time steps and controls the flow of information within the network.

Hidden state: A cell that is responsible for the encoding of the most recent time steps of the data.

Forget gate: This gate determines what information from the cell state should be discarded and what should be retained. It uses a sigmoid function to make a decision . Two inputs x(t), i.e., an input of schedule sequence at the particular current time from the WSN and h(t − 1), i.e., the previous cell output of schedule sequence from the WSN, are fed to the forget gate and multiplied with weight matrices followed by the addition of bias. The result is passed through an activation function which provides a binary output. If, for a particular cell state, the output is 0, then the piece of information is forgotten, and for output 1, the information is retained for future use for the input gate.

Input gate: This gate is responsible for deciding what new information should be added to the cell state. It uses another sigmoid function for this purpose. First, the information is regulated using the sigmoid function and the values to be remembered are filtered similar to the forget gate using inputs h(t − 1), i.e., an input of schedule sequence at the particular current time from the WSN and x(t), i.e., the previous cell output of schedule sequence from the WSN. Then, a vector is created using the tanh function that gives an output from −1 to +1, which contains all the possible values from h(t − 1), i.e., an input of schedule sequence at the particular current time from the WSN and x(t), i.e., the previous cell output of schedule sequence from the WSN. Finally, the values of the vector and the regulated values are multiplied to obtain useful information.

Output gate: This gate controls what information from the cell state should be used to produce the output. It employs a hyperbolic tangent (tanh) function to create a vector, which is then used for the prediction. First, a vector is generated by applying the tanh function on the cell. Then, the information is regulated using the sigmoid function and filtered by the values to be remembered using inputs h(t − 1), i.e., an input of schedule sequence at the particular current time from the WSN and x(t), i.e., the previous cell output of schedule sequence from the WSN. Finally, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell.

The training process of the LSTM involves three sigmoid functions and two hyperbolic tangent functions. These functions are well-suited for handling data that evolve over time [11]. The training of the LSTM network results in an equation with two variables, one of which is time. This enables the network to predict one variable when the other is provided. The accuracy of these predictions serves as a measure of the LSTM network’s efficiency.

In WSN applications, each round involves predicting the network configuration with a focus on optimising energy usage and minimising the distance from the sink. These network configurations are linked to the locations of nodes and play a crucial role in achieving convergence and extending the network’s lifespan.

7.6 Experimentations Set-Up

MATLAB simulator was employed to conduct simulations and experiments on the network. Specifically, we exclusively compared the LSTM scheduling algorithm with the SOFM algorithm. This selection is based on SOFM's demonstrated superiority over other algorithms that were previously investigated, namely HMM, and RCS. The SOFM node scheduling algorithm uses features to provide the schedule solution. There are five features considered in the process of SOFM, namely: (1) the position of cluster heads to the Sink, (2) the distance between the parent to child nodes, (3) the energy levels, (4) the Transmission link (Tx), and (5) the Receive link (Rx). In the SOFM process, the Euclidean distance is calculated, and the best matching is identified where the clustering starts to form a group as a potential solution. Each group have its scheduling which is based on a value obtained from the features. The SOFM algorithm provides a spatial customised cluster-based configuration that is associated with a certain weight value which is updated in every iteration. All these features’ processes are calculated by the BS during the design time and implemented in the run time. Thus, we find it suitable to benchmark the new proposed algorithm against SOFM for performance evaluations. The experiments were conducted using the following parameters, and all resulting data were collected over a 30-run period.

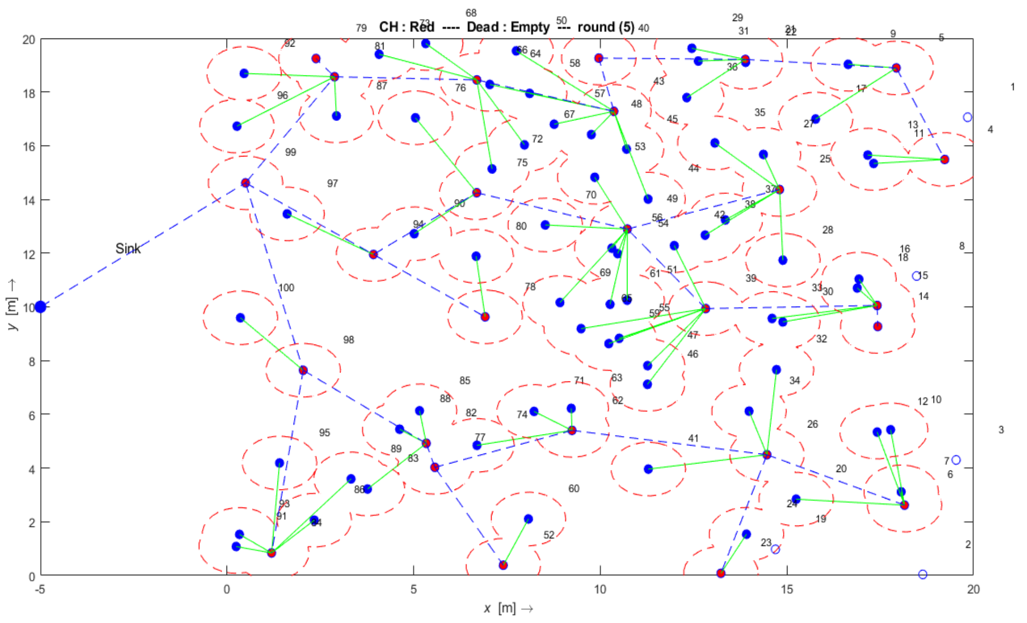


Figure 7.4 Simulation Set-up of LSTM Node Scheduling Algorithm

Figure 7.4 demonstrates the simulation experiment of the LSTM with 100 homogenous nodes i.e., having the same capabilities (sensing, communicating, power battery, processing power). The following assumptions were considered:

1. The WSN monitored area in simulations is 100 m^2.
2. All nodes are connected to one sink node.
3. Nodes are randomly deployed.
4. The number of iterations is fixed to 200 epochs.
5. The number of simulation iterations is limited by the convergence of the LSTM network when the wait time reaches a stable condition.

The Simulation experiment was individually tested 30 times as mentioned above using one-way ANOVA analysis. ANOVA is a method to analyse and test the variance of the average of one or more group of data. A one-way ANOVA analysis was implemented on the SOFM and the LSTM algorithms to identify the P-value for statistical results. Table 7.1 presents the statistical analysis of both LSTM and SOFM node scheduling algorithms.

Table 7.1 Statistical Analysis

Metrics	SOFM Algorithm	LSTM Algorithm
Number of Hops	28	50
Connectivity	0.99	0.99
Coverage	0.99	1.6
Lifetime	126	46.69
ANOVA 1	6.04755e-89	1.11e-16

1. 100 homogenous nodes i.e., having the same capabilities (sensing, communicating, power battery, processing power). The WSN monitored area in simulations is 100 m2.
2. All nodes are connected to one sink node.
3. Nodes are randomly deployed.
4. The number of iterations is fixed to 200 epochs.
5. Number of simulation iterations is limited by convergence of the LSTM network, when the wait time reaches a stable condition.

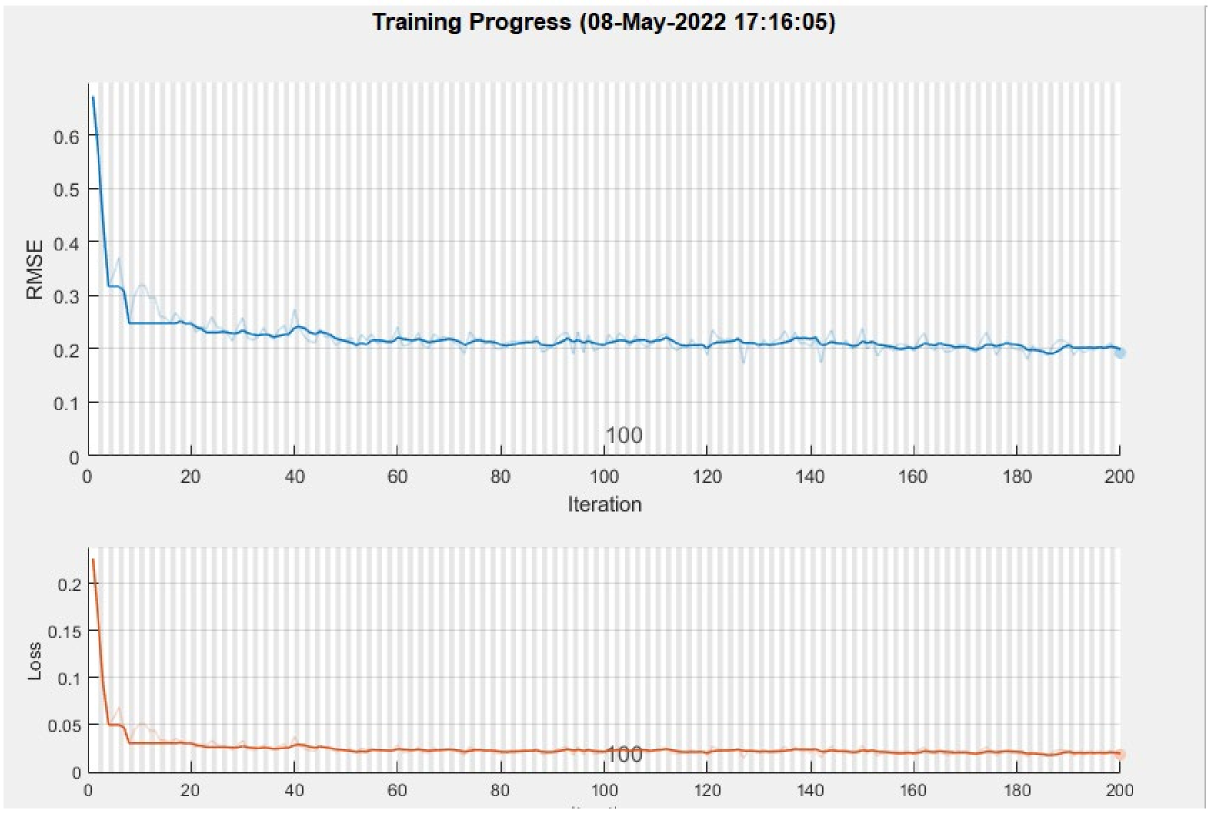


Figure 7.5 Training Progress

Our algorithm training is needed to identify the sequence of efficient coverage. As illustrated in Figure 7.5 above, the Root Mean Square Error (RMSE) shows a clear trend over iterations.

It reaches a saturation point after the 40th iteration, suggesting that the network weights have stabilised. This stabilisation is reflected in the improvement of the loss function, which in turn enhances the accuracy of the training.

7.6.1 All Used Energy

This metric refers to the energy used in the whole network throughout the simulation time until all nodes fully deplete/consume their energy. The X-axis represents time in round units, and the Y-axis is the energy used as shown in Figure 7.6.

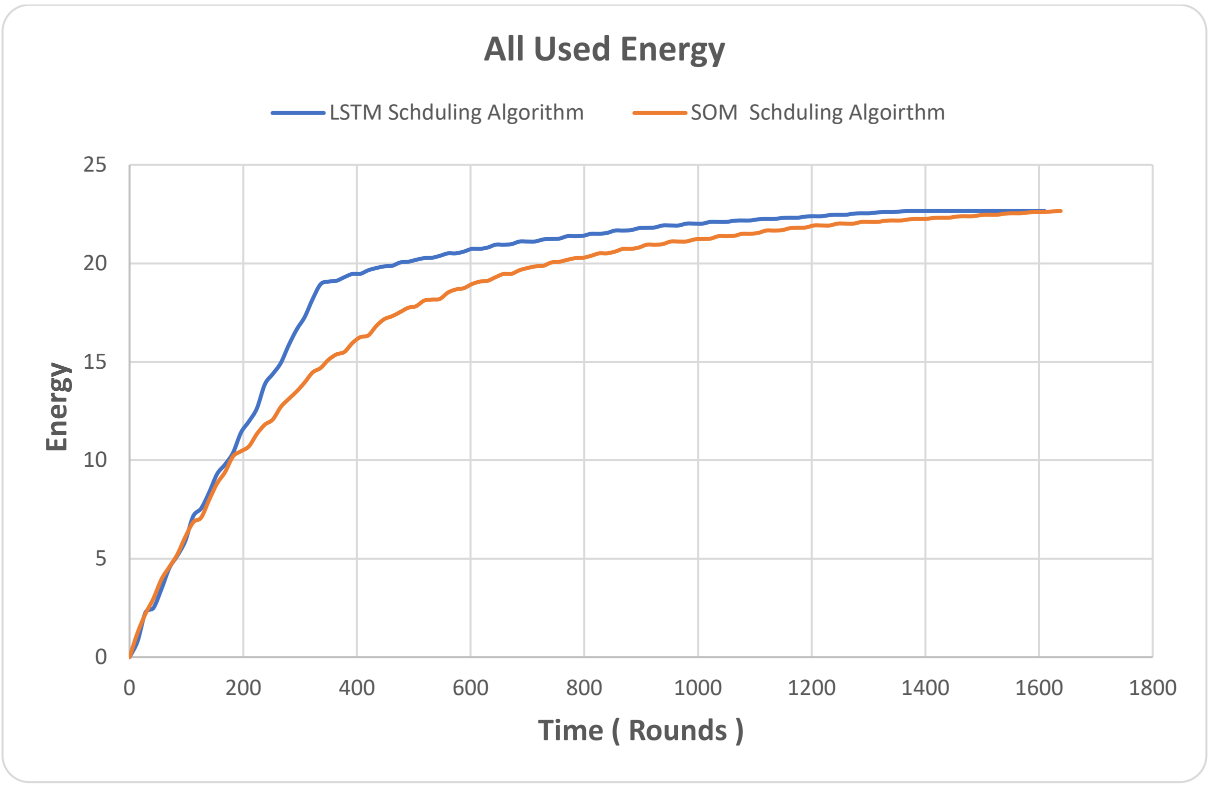


Figure 7.6 All Used Energy

Although both algorithms had a similar consumption rate of energy at the beginning of the simulation, up until closer to the 200th round, the SOFM algorithm consumed overall lower energy than the LSTM algorithm. However, the difference in consumption rate gets smaller and smaller after the 600th round, until towards the end of the simulation time when the difference is close to zero. This counterintuitive observation stems from the trade-off design in LSTM where we invoke the maximum ON nodes in the WSN. An important point to mention is that the dependability (e.g., service availability and reliability of the network) of WSN is more temporal by nature this is because the energy of the WSN depletes concerning time as compared to the spatial change in terms of geometry. This point is noted from the performance of the LSTM algorithm in terms of energy consumption as marked by the blue curve.

7.6.2 Lifetime of Sensor Nodes

This metric represents the number of live sensor nodes in the network during the simulation (network lifetime). This is shown as the Y-axis in Figure 7.7 below.

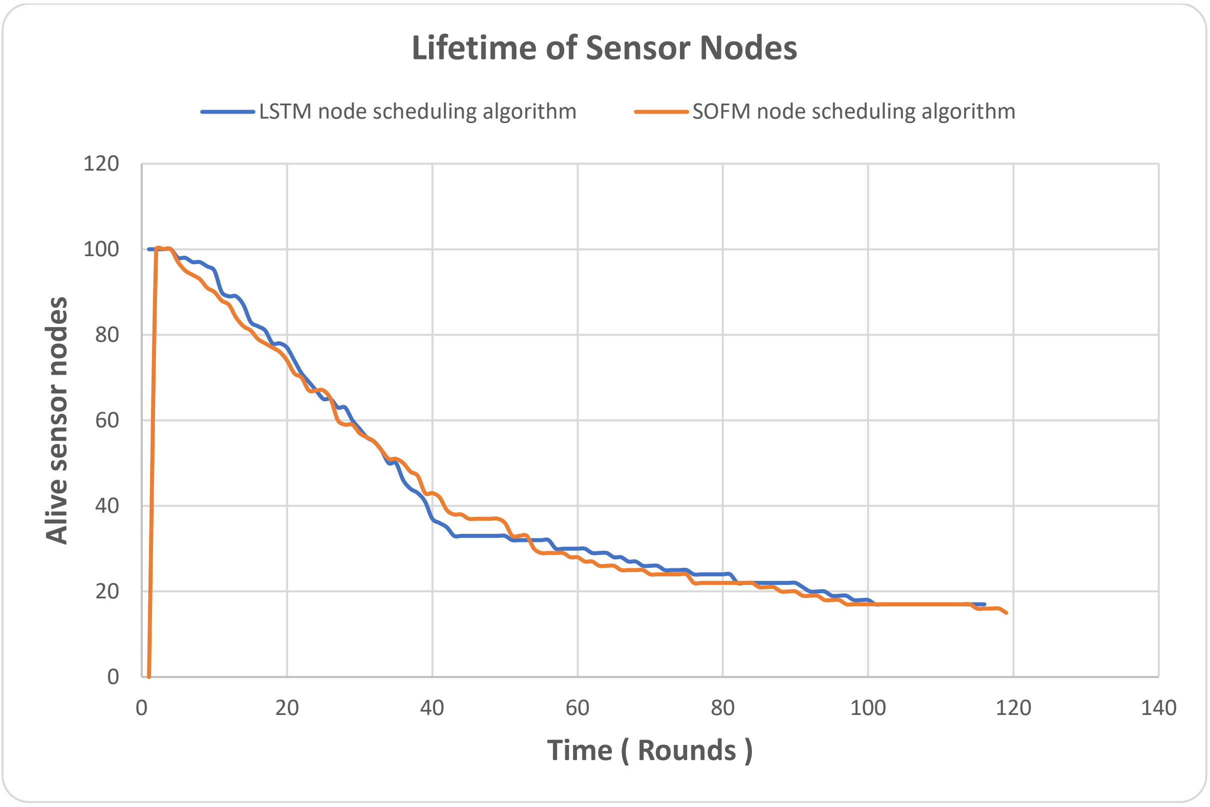


Figure 7.7 Lifetime of Sensor Nodes

The curves here affirm our observation in 7.6 where both algorithms kept the WSN alive for a similar number of simulation rounds. Since we need at least 2 input values at the time (t-1), t to predict the value at t+1, so the LSTM curve i.e., the blue curve starts from a gap. It is noted in the above figure that LSTM curve performed lower than SOFM exactly between 40 rounds to 55 rounds, this is due to the number of short sequences of nodes that could provide us with the best lifetime performance as well as coverage and connectivity. The results here support our hypothesis in finding a better scheduling algorithm for service availability without compromising other metrics drastically like this one where we can observe fewer Alive Sensor Nodes by LSTM than SOFM, however not much lower throughout the simulation time. There is not much improvement in the lifetime of the sensor network in the LSTM at the 117 rounds, case because the training network did not provide sufficient energy residues to carry for the next rounds. This is due to the threshold (this is the cut-off energy that the nodes will not continue operation) of the minimum energy setup in the sensor nodes in the WSN.

7.6.3 Efficiency

The bandwidth of the network is defined as the number of packets generated in the network, so the lower bandwidth will have less congestion and less information, and more bandwidths will result in more congestion and more information. The X-axis is the number of rounds, and the Y-axis is the number of packets in the network as shown in Figure 7.8 The main point of interest is the number of packets sent and received in the WSN.

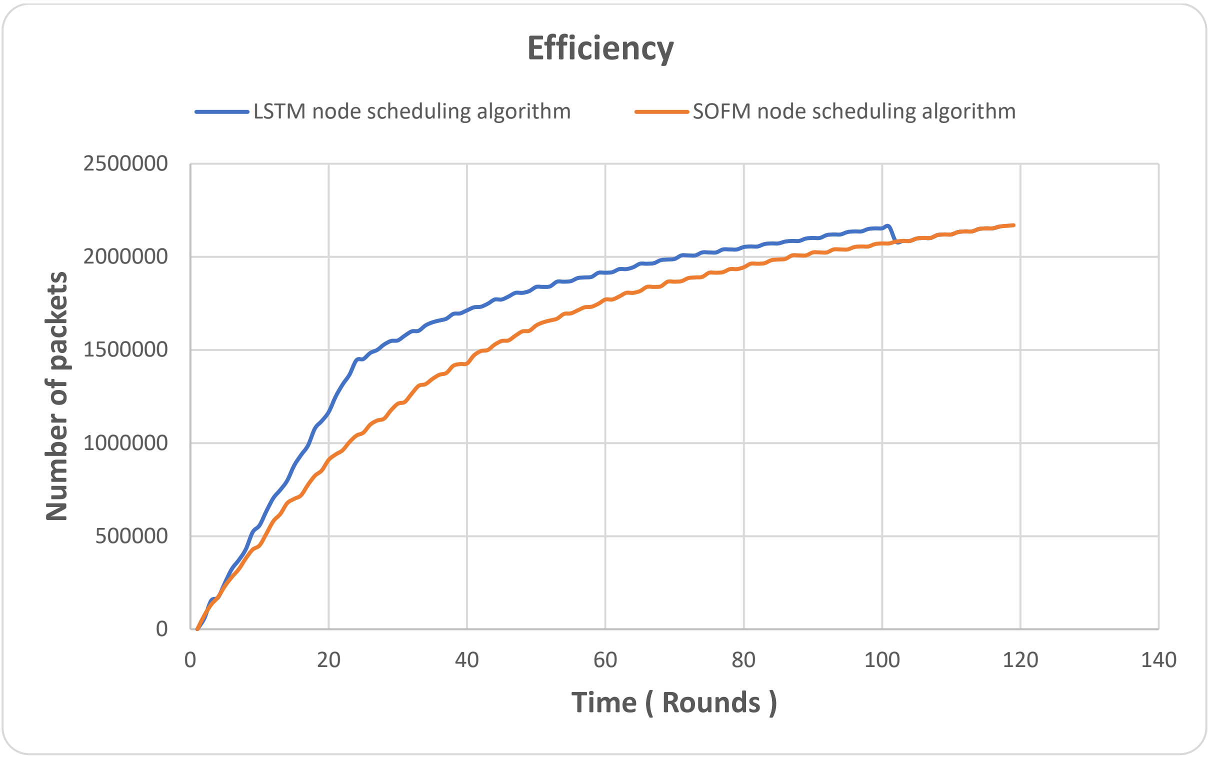


Figure 7.8 Efficiency.

For efficiency, The LSTM curve i.e., the blue curve is efficiently transmitting and receiving the information in the considered LSTM network as can be noted from the 10 to 105 rounds. The LSTM algorithm has shown its advantage over the SOFM algorithm in providing more efficiency up until the end of the simulation rounds where they were the same. It is also a usual observation for having both algorithms to have similar efficiency performance towards the end of the simulation because the number of alive nodes by then is less and similar, hence no matter how many data packets are generated constrained by the availability of alive nodes, the efficiency is the same. Again, we can notice here after 105 round there is a stoppage of the LSTM i.e., the blue curve, this is due to the threshold (this is the cut-off energy that the nodes will not continue operation) of the minimum energy setup in the sensor nodes in the WSN.

7.6.4 Connectivity

Connectivity is the measure of the number of nodes connected to the network. In Figure 7.9 Y-axis is the performance of connectivity which is measured by number as a value (fraction of nodes connected in the network). In our simulation we have 100 nodes in the network, if the connectivity is 0.98, then the total connected nodes are 100*0.98=98 and the X-Axis is the number of rounds.

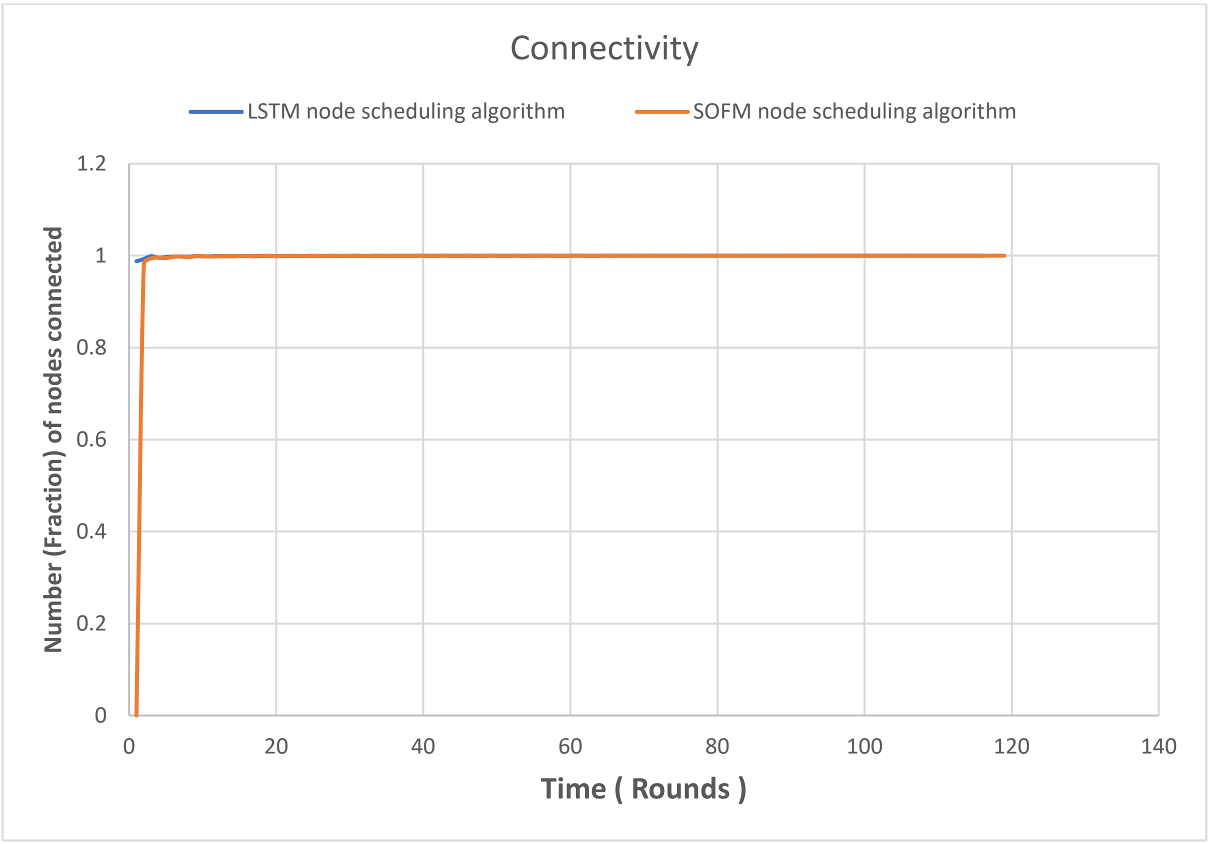


Figure 7.9 Connectivity.

From this figure 7.9, it can be concluded that in support of the thesis hypothesis, the newly proposed LSTM algorithm didn’t compromise or outperform the SOFM algorithm in connectivity. LSTM focused on improving service availability without compromising the other metrics e.g., efficiency, coverage, and lifetime of sensor nodes. In LSTM, some metrics like the lifetime of sensor nodes, efficiency and coverage outperformed or remained the same with small variations as SOFM due to efficient temporal management of energy and topology of the WSN, however, in connectivity we did not observe much difference between the two algorithms which resides in the remit of our hypothesis. Notably, the blue curves starting for sensor networks are not positively controlled in terms of messages sent and revised by the Base station but rather indirectly managed which is one of the reasons why these networks are prone to perturbations.

7.6.5 Coverage

Coverage is the measure of the test field of interest. In Figure 7.10, the Y-axis is the performance of coverage which is measured by number as a value (fraction of nodes connected in the network). In our simulation, the coverage is represented by the fraction of the area covered by the network. If the area of the test field is 100 m2, the coverage of 0.8 would mean 0.8*100 = 80 m2 has been covered. The X-axis is the number of simulation rounds.

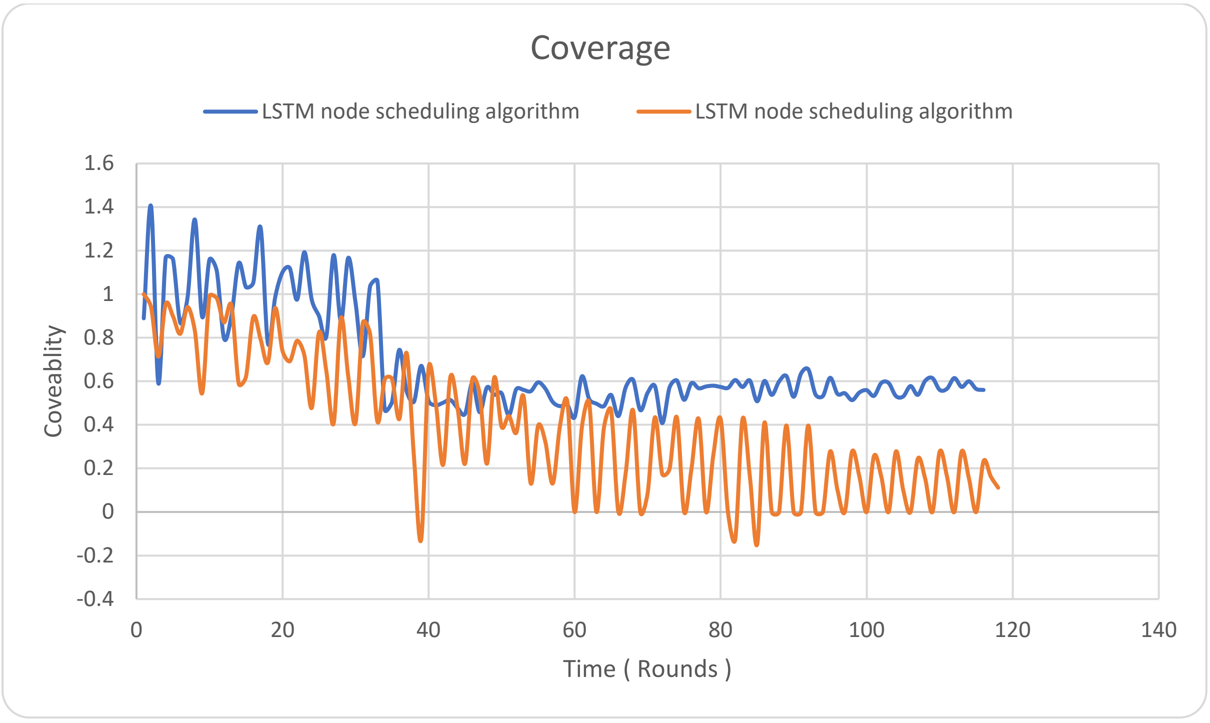


Figure 7.10 Coverage.

The main objective here is to achieve the total coverage of all ON nodes (i.e., the sequence of efficient nodes a stable coverage (when fluctuations are low in the Y axis)) to be greater than or equal to the area to be monitored. Hence, there is need to stabilize the coverage and optimize it to acceptable network lifetime. It can be observed that from the 39th round, the coverage starts to converge to a less fluctuating state up until the 116th round. Our hypothesis is to find and detect a sequence of efficient coverage and then replicate it using LSTM throughout the entire network life cycle. The Figure illustrates how our hypothesis is achieved as the LSTM values are more stable towards the second half of the simulation and are also higher in value than SOFM.

7.7 Evaluation and Interpretation

The hypothesis is proved by finding an efficient sequence of coverage with improved coverage and without compromising network connectivity and lifetime. Our previous algorithms have uncovered multiple issues in scheduling algorithms. One issue was the fluctuating behaviour in coverage - with LSTM, this was managed to be a more stable behaviour after the first half of the network lifetime after the LSTM network reached a certain acceptable level of accuracy. More training data will require more time for training and less data will require less amount of training time for training. There is a trade-off; when more time is utilised, the output is more accurate than when less time is utilised. These findings are beneficial for safety critical WSNs which are the area of research interest by the author of this thesis, where the coverage has been observed to be higher when using the LSTM algorithm than the SOFM node scheduling algorithm. There is also another benefit for the LSTM node scheduling which is manifested in the automation of its design process: LSTM first spends its efforts and time in finding a good sequence of coverage. For example, this is why the start of the coverage curve in Figure 7.9 was fluctuating), however once the good values are identified, LSTM automatically ensures coverage for the rest of the network lifetime there will be guarantees for service availability and reliability.

The efficiency of the LSTM algorithm was obtained using epochs (iterations) to confirm our desired level of quality-of-service levels. For example, when training is performed, functions like loss and error, both need to be on scoring low in (i.e., minimizing) the Root Mean Square Error (RMSE) and Loss Function as illustrated in Figure 7.5, this is performed using 200 iterations in our experiments to reach a stable weight state. From the training graph in Figure 7.5 we can see that after 100 iterations both RSME and Loss Function score similar values with similar patterns of fluctuations throughout the remainder of the iterations. This indicates a stable configuration of the weights in the LSTM algorithm in case the stability is not achieved then the number of iterations must be increased to improve the training network.

In the context of future research considerations, we recommend refining the design of the LSTM algorithm to better handle training with a larger dataset, as the current algorithm encountered difficulties after processing just 100 training samples [Cheng et al., 2019]. This limitation is also applicable to large-scale WSNs containing millions of sensor nodes, where the LSTM’s memory constraints impede its performance. While transformers present a logical alternative to LSTM, our system’s capacity was limited to 100 to 150 nodes, and to reduce computational costs, we opted for LSTM.

Chapter 8 Conclusions and Future Work

8.1 Evaluation

This section presents a critical evaluation and comparison of the proposed solutions with recent existing published work in the literature. Overall, most Node Scheduling Algorithms in the literature do not address the three objectives (coverage, connectivity, and network lifetime) of the WSN altogether. In fact, most works address each objective in isolation, such as [Ambrose, 2008, 2008; Cerpa and Estrin, 2004; Chipara, 2010; Ha, 2006; He et al., 2004; Tate and Bate, 2011; Viera et al., 2003; Wu et al., 2005; Xu et al., 2001; Yang, 2011] and [Ye et al., 2003] do not consider network coverage in their works; while other works such as [Tian and Georganas, 2002] and [Cardei et al., 2005] consider network coverage but without ensuring good level of network connectivity; other work as [Cardei et al., 2005] and [Srivastava and Hasan, 2018] uses a very simple WSN scenario lacking tests on realistic and complex WSNs. In addition, [Vashisht, 2023] and [Rudati et al., 2023] work focuses on only to improve the network lifetime of the sensor nodes, without considering the performances of other objectives such as coverage, connectivity.

In [Bajaber, 2023] the solution address a partial coverage of the WSNs and lifetime. Alternative work of [Jaiswal and Anand, 2024; Pandiyaraju et al., 2023a; Shagari et al., 2020] address connectivity, coverage and lifetime in homogenous network scenarios. In [Harizan et al., 2024], uses Relay Nodes to provide better coverage connectivity and lifetime. Our investigation revealed that a few published works address the three objectives collectively in the node scheduling approach: the RCS algorithm by [Liu et al., 2006] and a Clique-based node scheduling by [Wang et al., 2010] solved the MOO using node scheduling algorithm.

There are several approaches that utilise ANN for different optimization problems in field of WSNs. For an example, the approach in [Masti et al., 2019] utilises the ANN to discover patterns that are of use for WSNs activities such as node scheduling, energy consumption, packet communications for virtual sensors design irrespective of energy, coverage and connectivity considerations. Another example presented in [Sharma and Kansal, 2023] that uses ANN in the cluster head selection mechanism to enhance the network lifetime in WSNs. On the other hand, [Vidhya, 2023] proposed an ANN method that trains the network on a large dataset, enabling it to adapt to various scenarios and extend the network's lifetime. The proposed method is limited to a small scale WSN. In contrast, in our study, the proposed ANN is built for analysing the node scheduling in WSNs and investigating its effects on key performance metrics such as network lifetime, coverage and connectivity simultaneously and in large scale WSN. Performing manual analysis is a difficult task, especially in a large distributed WSN system. Hence, necessitate the need to automate the analysis using the ANN. Unlike the approach proposed by [Vidhya, 2023] and [Sharma and Kansal, 2023] where the work is targeting the routing protocol at the network layer. In this thesis, the ANN is used to analyse the complex node scheduling behaviour in WSNs at the level of application layer, where key parameters such as positional distances of each node from the Base station, residual energy and communication range of each node are considered to reveal certain potential areas of improvement in the RCS Node Scheduling Algorithm.

Given the ANN analysis of the node scheduling behaviour, a limitation on which nodes will be overworked e.g., consume energy faster are obtained and classified. The HMM model has been used widely in the field of the WSN but in a different context. For example, [Krishnamurthy, 2002] addressed the problem of data reliability by obtaining certain physical data e.g. measurements of certain processes, known as signal processing applications. However, there is an interesting example in the literature that fits within the context of this study, [Qihua et al., 2015], that used the HMM to model the behaviour of the WSN. In this thesis a new HMM node scheduling algorithm is introduced and compared against the RCS in [Liu et al., 2006]. In the HMM node scheduling algorithm, the optimal solution is achieved by ensuring that each node is used for transmission dependent on its current energy level. As the latter decreases, the probability for the node to be in a transmit state decreases while its probability of being in a receive state increases. Thus, the new solution improved the network lifetime while maintaining the required level of connectivity and coverage of the WSN.

Although the HMM Node Scheduling Algorithm has shown an improvement over the existing RCS algorithm in the literature, it is limited to the information made available by the sensor node e.g., stuck in local maxima and local minima. Hence, a Bio-inspired Bat algorithm has been proposed to find an optimal solution within its search space. There are various Bio-inspired algorithms for different MOO problems in WSNs. For instance, [Jaiswal and Anand, 2024] uses the FA to achieve target location coverage and sensor node connectivity. However, the study in [Jaiswal and Anand, 2024] did not cater for heterogeneity which is an important mechanism for a resource constrain WSN. Another limitation in [Jaiswal and Anand, 2024] is that the entire approach is based on target-based coverage calculation. Another example for the use of the Bio-inspired approach in WSNs is proposed by [Ovabor and Atkison, 2024], where a hybrid MOO technique integrates cascading firefly and Greywolf algorithms to improve connectivity, coverage and network lifetime in WSNs. The difficulty of firefly and Greywolf type of hybrid optimisation technique is in the increase of computational complexity and more convergence time required for optimization. Another recent approach proposed by [Yang and Xia, 2024] which addresses the coverage via the use of dynamic cluster routing on a non-uniform clustering type of WSNs. The authors in [Yang and Xia, 2024] present a very unrealistic view of energy loss of less than 1% which is the case if fewer packets are delivered and almost no control packets are generated in this case.

Accordingly, a Bio-inspired computation opened the way for search space exploration; hence a Bat-based scheduling algorithm is proposed to determine if a solution better than that proposed by an earlier HMM algorithm is possible. The Bio-inspired Bat node algorithm generates a set of solutions randomly and then uses a loop search to find the optimal solution based on an objective and fitness functions. In comparison to the recent discussed approach in the existing literature i.e., [Jaiswal and Anand, 2024], [Ovabor and Atkison, 2024], and [Yang and Xia, 2024], where the technique and the interest is different. Thus, the results of the Bat node scheduling algorithm in comparison to the HMM showed significant improvements on all three objectives (coverage, connectivity, and network lifetime) for a given search space. This leaves open the possibility for another algorithm to outperform our Bat algorithm. Hence, exploring a new state space necessitates a new and different representation of the problem domain.

Subsequently, an interesting observation of the coverage objective have led to the utilization of SOM model to represent the search space from a spatial perspective in WSN. The spatial features of the SOM enabled a better abstract representation based on special configuration.

Similar to the HMM and Bat models, the SOM model has also been used widely in the literature though in different context. For instance, the work in [Kulkarni et al., 2021] used Self-organising map-based for dynamic decision-making algorithm for node deployment for heterogeneous WSN. The author in [Kulkarni et al., 2021] addressed the coverage hole problem using node mobility. The solution to the problem is practically not feasible as in remote environments the nodes cannot be moved through locations unless it is mounted on some vehicle. Another example that used the SOM model is [Mittal and Kumar, 2015]. The solution tends to reduce the data communication during the entire network lifetime by reducing its size. In this work [Mittal and Kumar, 2015], 1500 data volumes were generated and aggregated from all sensor nodes and sent to the base station. the real mapping of the coverage is absent in this case, it can be used for data transferring only. Another interesting study that used the SOM model is [Talei et al., 2023] to optimize and use energy efficiently to be used in HVAC indoor systems. This study considered energy optimisation and it is used in a very specific context.

Hence, an SOFM node scheduling algorithm is proposed to provide a better performance solution with respect to all three objectives of the MOO problem. In particular, the self-organising capacity in the SOFM algorithm outperforms the Bat node scheduling algorithm through its simplistic method of dimension reduction and classification. The SOFM results indicated a good sequence of spatial coverage which could be captured for further improvement. Accordingly, to further fulfil this objective, the problem domain is represented from a spatial to a temporal state space where the LSTM model is proposed.

The LSTM model is especially designed to account for the temporal data types which is highly suitable for data analysis and prediction modelling in WSN. The main idea behind using LSTM lies in the training of the data generated by the WSN system. In other words, the longer the data the better the prediction in the time series of WSN parameters like connectivity, coverage, and lifetime. The use of LSTM for WSN systems is appropriate as the network parameters tend to be nonlinear by nature, which is very hard to model. Hence, data abstraction e.g., wrangling, and crushing methods are used by different LSTM layers coupled will activation function like sigmoid, tanh etc. The LSTM model, provide highly optimised coverage and connectivity and lifetime with the available physical parameters (i.e., residual energy, position of the node in the WSN).

The LSTM model is used in WSNs but in specific optimization contexts, such as the work in [Cheng et al., 2019] which aims to reduce unnecessary data transmission thereby improving the quality of the data collected in WSNs. The approach in [Cheng et al., 2019] did not cater for the connectivity, coverage and a lifetime of the network. [Mohanty et al., 2020] attempted to reduce energy consumption by utilising RNN-LSTM based models, but their approach is limited to communication reduction only, thereby, improving the network lifetime. Another interesting example in [Jeyalaksshmi and Ganesh, 2022] utilised the Bidirectional Long Short-Term Memory (BiLSTM) technique and provided adaptive duty-cycle scheduling for WSNs;

they address the energy consumption to improve network lifetime problem, but fail to cater for the coverage and connectivity issues. In this thesis, the LSTM node scheduling algorithm exploits spatial-temporal correlations among sensory data by tuning the available physical parameters to address the best coverage, connectivity, and lifetime, which is not addressed by a single paper till date.

8.2 Conclusion and Future Work

This thesis addressed the MOO problem in the domain of safety-critical systems in the WSNs. The principles of WSN dependability, service availability and reliability were the main considerations in this work. The motivation of this work emerged from, failure related to, the limited resources of the WSN systems, i.e., processing, memory capacity and mainly limited battery life. Hence improving the energy consumption would in turn improve the service availability and reliability. The solutions proposed are based on novel scheduling frameworks utilising methods such as ANN, HMM, BAT, SOFM and LSTM models to find a justifiable optimal solution. Each proposed solution adopts a different representation of the problem to obtain better results in the state space which reduce the computational complexity along with the improvement of connectivity, coverage, and energy lifetime. First, in order to build this research on a solid foundation an existing and proven approach from the literature was replicated with validation and verification activities.

This work proposed six main contributions that each was discussed in a separate chapter, in which, each contribution addressed each research question distinctively. Accordingly, there **are six research questions** which are derived from the research hypothesis proposed in Section 1.3. The first research question is whether the ANN approach can be used in safety critical WSNs to find suitable scheduling solutions for MOO problems. In Chapter 3, the ANN Analyser algorithm chapter, the first research question was addressed with the proposed ANN Analyser of the RCS algorithm to analyse its scheduling performance and demonstrate improvements with respect to the MOO problem in WSNs. The solutions stemming from such analysis show improved performance than existing policies in the literature. In particular, the New ANN algorithm was able to detect the “Overused” Nodes e.g., the number of times nodes were switched ON and the “Underused” nodes e.g., the number of times nodes were switched OFF during the run time of the simulation. Then classify the nodes that are "Overused" into a Class 2 otherwise 1. When balancing the workload or the sleep schedule of those nodes, the network's lifetime increases by avoiding unnecessarily depleting nodes' energy. Those nodes tend to be, as found in this work, closer to one or two Hops from the Base station and naturally experience a high workload. Thus, saving energy for these critical nodes would not only increase the network's lifetime but also increase its dependability for safety-critical applications. Given the result of the ANN analyser, it would be interesting to synthesize a new scheduling that would improve on the scheduling from the analysed algorithm (here RCS) or would return a verdict indicating that no further improvement is possible. This question is considered for future work. Alternatively, a new scheduling algorithm could be proposed "manually"; however, approaches different from that used by RCS exist which can provide better results such as the Hidden Markov Model-based algorithm proposed.

The second research question is whether (HMM) can provide performance improvements for scheduling algorithms to address the MOO problem. In Chapter 4, this research question was addressed by introducing a novel HMM algorithm, which proposes a scheduling algorithm that is based on Transition Probabilities (ref w.r.t transition probabilities), unlike the original RCS which is based on random scheduling. The HMM improved the network lifetime while maintaining both connectivity and coverage of the network. A scheduling algorithm was introduced for the node duty cycle e.g., the manipulation of transmission and receiving states. The transition states of the nodes are restricted to four states in the algorithm to avoid computational overheads. The main goal here is to determine the receive and transmit states' schedule of each node in the WSN in the design time. As a result, each node knows it receive schedule and operates accordingly in the runtime. In this process, it intervened in the transitioning probability between the states that are received and transmit states. Network connectivity and path optimality are important metrics considering broadcast collision and channel errors as quality-of-service metrics. The RCS algorithm has had several errors due to inefficient network configuration (e.g., The spatial network topology was not in favour of better data transmission), but it was mitigated using transition probability-based optimization to obtain better network configuration in the HMM algorithm.

The third research question is Bio-inspired node scheduling mechanism can provide further performance improvements on existing scheduling algorithms in the literature to address the multi-objective WSN optimization. In Chapter 5, Bio-inspired computation was demonstrated to address the M-covered connected optimization problem in WSNs [Liu et al., 2006]. This was achieved by the proposed Bat algorithm that is coupled with the Pareto optimisation algorithm which has improved the performance of the network parameters like connectivity, coverage, and efficiency. The network lifetime, coverage, and connectivity were improved and contributed to the dependability of WSNs where assumptions/simulation parameters were used to make the work in this thesis close to real-time WSN scenarios. The specification of the Bat algorithm improved energy and distance which were not accommodated in other algorithms were included in the simulation experiments. Distance is interpreted as connectivity, and energy is referred to as network lifetime.

The fourth research question is about further improvement in scheduling performance when using SOFM by implementing a self-organizing map to reduce the data from a higher dimension to a lower dimension to further improve the scheduling multi-objective optimisation performance. In Chapter 6, there was a further improvement by introducing on-service availability, and reliability compared to the Bat algorithm (which in turn was shown to be better than the previously proposed algorithms in this thesis). The self-organising capacity in the SOFM algorithm outsmarts the Bat algorithm through its simplistic method of dimension reduction and classification. In particular, the SOFM algorithm has outperformed the Bat algorithm in network lifetime, where in some metrics both have scored similar values, however, the SOFM algorithm always provides an extended network lifetime over the Bat algorithm. The cost of implementing the BAT algorithm is higher than the SOFM algorithm. In the Bat algorithm, there was a complexity level with respect to the operational design and fitness and objective functions settings coupled with the Pareto algorithm to select the best possible solution which was higher than the SOFM set-up. The SOFM is a neural network-based dimensionality reduction algorithm.

The fifth research question will show an algorithm that can identify and remember the good coverage states to help further improve service availability without compromising network lifetime. In Chapter 7, an efficient sequence of coverage based on LSTM is proposed with improved coverage without compromising network connectivity and lifetime. One issue, in particular, can be foreseen to be the fluctuating behaviour in the coverage metric. The findings of this chapter can be beneficial for safety critical WSN systems which are the area of research interest.

The recommended future work of this thesis will revolve around the possibility of a framework that can support the decision-making for achieving a minimum, acceptable level of coverage and connectivity with minimal ON nodes. This will be achieved by a testing framework that helps in identifying and evaluating the total coverage percentage per zone to the total WSN. Therefore, a novel testing framework will be proposed to test the performance of WSN scheduling algorithms. The new testing framework will identify the number of nodes in the WSN then it will classify and prioritize them based on their positions/coordinates from the Base Station. A step-side refinement approach will be used and the WSN into a number of zones, where, in each zone, each node's performance will be tested with respect to coverage, connectivity and network lifetime. Then each zone will be examined with respect to the network's total coverage, connectivity, and network lifetime. This way, the weakness of the algorithm can be identified. The proposed testing framework will provide a key design principle to better analyse and propose a scheduling algorithm that meets the QoS requirements with respect to coverage, connectivity, and network lifetime in safety critical WSNs. The future new testing framework shall test the behaviour of the WSN and the performance of the WSN scheduling algorithm, during the run time, based on the main factors below:

- Number of zones, where a WSN is divided into a number of zones based on their positions from the Base station. E.g., 1 Hop zone, 2 Hops zone, etc.
- Number of Hops (the connection between two nodes is called Hops). The network is required to be covered by a number of hops to connect the outlier nodes.
- Average nodes per zone, the average number of nodes per zone.

Additionally, the future testing algorithm shall uncover multiple issues in scheduling algorithms. One issue, in particular, is the fluctuating behaviour in coverage. These findings will be beneficial for safety critical WSNs which are the area of research interest in this research work. Also, evaluating the introduction of mobile nodes is recommended that act as backup nodes to avoid network partitioning and improve the network's lifetime in the realm of WSN scheduling. In addition, the testing framework can be automated with ML functions after testing any scheduling algorithms and automatically lay out the number of zones and coverage performance as well as connectivity with the desired network lifetime. Furthermore, the simulation project can be emulated in real life by considering a test bed of 20 real sensor nodes which can be replicated for 100 to 150 nodes in software, the simulation results can be tested on this test bed using a customised WSN framework that uses Message Queue Telemetry Transport (MQTT) protocol for data transfer – MQTT is an OASIS standard messaging protocol for the Internet of Things (IoT).

Reference

Abdulzahra, A.M.K., Al-Qurabat, A.K.M., Abdulzahra, S.A., 2023. Optimizing energy consumption in WSN-based IoT using unequal clustering and sleep scheduling methods. Internet of Things 22, 100765.

Adday, G.H., Subramaniam, S.K., Zukarnain, Z.A., Samian, N., 2024. Investigating and Analyzing Simulation Tools of Wireless Sensor Networks: A Comprehensive Survey. IEEE Access 12, 22938–22977. https://doi.org/10.1109/ACCESS.2024.3362889

Aliyu, F., Umar, S., Al-Duwaish, H., 2019. A survey of applications of artificial neural networks in wireless sensor networks, in: 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO). IEEE, pp. 1–5.

Alrajeh, N.A., Lloret, J., 2013. Intrusion Detection Systems Based on Artificial Intelligence Techniques in Wireless Sensor Networks. International Journal of Distributed Sensor Networks 9, 351047. https://doi.org/10.1155/2013/351047

Ambrose, A.I., 2008. Scheduling for Composite Event Detection in Wireless Sensor. Master thesis, Florida Atlantic University.

Ammari, H.M., Das, S.K., 2006. Coverage, connectivity, and fault tolerance measures of wireless sensor networks, in: Symposium on Self-Stabilizing Systems. Springer, pp. 35–49.

Arrobo, G., 2012. Improving the Throughput and Reliability of Wireless Sensor Networks with Application to Wireless Body Area Networks.

Aslan, Y.E., Korpeoglu, I., Ulusoy, Ö., 2012. A framework for use of wireless sensor networks in forest fire detection and monitoring. Computers, Environment and Urban Systems 36, 614–625.

Avizienis, A., Laprie, J.-C., Randell, B., others, 2001. Fundamental concepts of dependability. University of Newcastle upon Tyne, Computing Science.

Bai, R.G., Qu, Y.G., Zhao, B.H., Lin, Z.T., Yang, C.M., 2008. Intelligent Monitoring Scheduling in Wireless Sensor Networks, in: 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing. IEEE, pp. 1–4.

Bajaber, F., 2023. Node Scheduling Scheme for Wireless Sensor Networks with Partial Coverage. Journal of Telecommunication, Electronic and Computer Engineering (JTEC) 15, 1–7.

Bhatia, V., Jaglan, V., Kumawat, S., Kaswan, K.S., 2019. A hidden markov model based prediction mechanism for cluster head selection in WSN. International Journal of Advanced Science and Technology 28, 585–600.

Busse, M., 2008. Algorithms for Energy Efficiency in Wireless Sensor Networks. Universität Mannheim 1–206.

Cardei, M., Thai, M.T., Li, Y., Wu, W., 2005. Energy-efficient target coverage in wireless sensor networks, in: INFOCOM 2005. 24th Annual Joint Conference of the Ieee Computer and Communications Societies. Proceedings Ieee. IEEE, pp. 1976–1984.

Cerpa, A., Estrin, D., 2004. ASCENT: Adaptive self-configuring sensor networks topologies. mobile computing, IEEE transactions on 3, 272–285.

Chawla, M., Duhan, M., 2015. Bat algorithm: a survey of the state-of-the-art. Applied Artificial Intelligence 29, 617–634.

Chen, M.-T., Chen, M.-H., Sun, C.-W., Cheng, Y.-J., Chou, C.-F., 2014. Joint Cluster Routing and Sleep-Awake Scheduling for Energy-Efficiency in Wireless Sensor Networks, in: 2014 IEEE International Conference on Internet of Things (iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom). IEEE, pp. 526–531.

Cheng, H., Xie, Z., Wu, L., Yu, Z., Li, R., 2019. Data prediction model in wireless sensor networks based on bidirectional LSTM. EURASIP Journal on Wireless Communications and Networking 2019, 1–12.

Chiang, M.-H., 2007. Energy Optimization in Sensor Networks. ProQuest.

Chipara, O., 2010. Towards real-time wireless sensor networks.

Cordina, M., Debono, C.J., 2008. Increasing wireless sensor network lifetime through the application of SOM neural networks, in: 2008 3rd International Symposium on Communications, Control and Signal Processing. IEEE, pp. 467–471.

Dener, M., Al, S., Orman, A., 2022. STLGBM-DDS: An Efficient Data Balanced DoS Detection System for Wireless Sensor Networks on Big Data Environment. IEEE Access 10, 92931–92945.

Forestiero, A., Pizzuti, C., Spezzano, G., 2009. Flockstream: a bio-inspired algorithm for clustering evolving data streams, in: 2009 21st IEEE International Conference on Tools with Artificial Intelligence. IEEE, pp. 1–8.

Goudarzi, R., Jedari, B., Sabaei, M., 2010. An efficient clustering algorithm using evolutionary hmm in wireless sensor networks, in: 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. IEEE, pp. 403–409.

Ha, R.W.K., 2006. A Sleep-Scheduling-Based Cross-Layer Design Approach for Application-Specific Wireless Sensor Networks.

Hameed, M.K., Idrees, A.K., 2024. Energy-aware scheduling protocol-based hybrid metaheuristic technique to optimize the lifespan in WSNs. J Supercomput 80, 12706–12726. https://doi.org/10.1007/s11227-024-05921-4

Harizan, S., Kuila, P., Kumar, A., Khare, A., Choudhary, H., 2024. Multi-objective Evolutionary Algorithms for Coverage and Connectivity Aware Relay Node Placement in Cluster-Based Wireless Sensor Networks. Wireless Pers Commun 135, 979–1008. https://doi.org/10.1007/s11277-024-11100-8

Hartung, C., Han, R., Seielstad, C., Holbrook, S., 2006. FireWxNet: A multi-tiered portable wireless system for monitoring weather conditions in wildland fire environments, in: Proceedings of the 4th International Conference on Mobile Systems, Applications and Services. ACM, pp. 28–41.

He, T., Krishnamurthy, S., Stankovic, J.A., Abdelzaher, T., Luo, L., Stoleru, R., Yan, T., Gu, L., Hui, J., Krogh, B., 2004. Energy-efficient surveillance system using wireless sensor networks, in: Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services. ACM, pp. 270–283.

Hidden Markov model states and emissions - MATLAB hmmgenerate - MathWorks India [WWW Document], n.d. URL https://in.mathworks.com/help/stats/hmmgenerate.html (accessed 6.29.24).

Hill, J.L., 2003. System architecture for wireless sensor networks. University of California, Berkeley.

Horneber, J., Hergenröder, A., 2014. A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks. IEEE Communications Surveys & Tutorials 16, 1820–1838. https://doi.org/10.1109/COMST.2014.2320051

Hu, P., Zhou, Z., Liu, Q., Li, F., 2007. The HMM-based modeling for the energy level prediction in wireless sensor networks, in: 2007 2nd IEEE Conference on Industrial Electronics and Applications. IEEE, pp. 2253–2258.

Huang, Y., Zheng, S., Zhang, P., Qin, X., 2010. Adaptive genetic simulated annealing algorithm for WSNs, in: Computer Engineering and Technology (ICCET), 2010 2nd International Conference On. IEEE, pp. V4-328.

Jaiswal, K., Anand, V., 2024. ESND-FA: An Energy-Efficient Scheduled Based Node Deployment Approach Using Firefly Algorithm for Target Coverage in Wireless

Sensor Networks. Int J Wireless Inf Networks 31, 121–141. https://doi.org/10.1007/s10776-024-00616-2

Jeyalaksshmi, S., Ganesh, R.M., 2022. Adaptive Duty-Cycle Scheduling using Bi-Directional Long Short-Term Memory (BiLSTM) for Next Generation IoT Applications, in: 2022 International Conference on Computing, Communication and Power Technology (IC3P). IEEE, pp. 75–80.

Jia, R., Zhang, H., 2024. Wireless Sensor Network (WSN) Model Targeting Energy Efficient Wireless Sensor Networks Node Coverage. IEEE Access.

Jin, S., Zhou, M., Wu, A.S., 2003. Sensor network optimization using a genetic algorithm, in: Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics. pp. 109–116.

Kim, J., Yoo, Y., 2020. Sensor node activation using bat algorithm for connected target coverage in WSNs. Sensors 20, 3733.

Knight, J.C., 2002a. Safety critical systems: challenges and directions, in: Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference On. IEEE, pp. 547–550.

Knight, J.C., 2002b. Safety critical systems: challenges and directions, in: Proceedings of the 24th International Conference on Software Engineering. pp. 547–550.

Kohonen, T., 2013. Essentials of the self-organizing map. Neural networks 37, 52–65.

Krishnamurthy, V., 2002. Algorithms for optimal scheduling and management of hidden Markov model sensors. IEEE Transactions on Signal Processing 50, 1382–1397.

Kubat, M., 1999. Neural networks: a comprehensive foundation by Simon Haykin, Macmillan, 1994, ISBN 0-02-352781-7. The Knowledge Engineering Review 13, 409–412.

Kulkarni, U.M., Kenchannavar, H.H., Kulkarni, U.P., 2021. Self-organising map-based dynamic decision-making algorithm for heterogeneous wireless sensor network. International Journal of Parallel, Emergent and Distributed Systems 36, 312–334. https://doi.org/10.1080/17445760.2021.1879069

Liu, C., Wu, K., Xiao, Y., Sun, B., 2006. Random coverage with guaranteed connectivity: joint scheduling for wireless sensor networks. IEEE Transactions on Parallel & Distributed Systems 562–575.

Masti, D., Bernardini, D., Bemporad, A., 2019. Learning virtual sensors for estimating the scheduling signal of parameter-varying systems, in: 2019 27th Mediterranean Conference on Control and Automation (MED). IEEE, pp. 232–237.

Meier, A., 2009. Safety-Critical Wireless Sensor Networks. SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH.

Miljković, D., 2017. Brief review of self-organizing maps, in: 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, pp. 1061–1066.

Mini, S., Udgata, S.K., Sabat, S.L., 2012. *M*-Connected Coverage Problem in Wireless Sensor Networks. International Scholarly Research Notices 2012.

Mittal, M., Kumar, K., 2015. Energy Efficient Homogeneous Wireless Sensor Network Using Self-Organizing Map (SOM) Neural Networks. African Journal of Computing & ICT 8, 179–184.

Mohanty, S.N., Lydia, E.L., Elhoseny, M., Al Otaibi, M.M.G., Shankar, K., 2020. Deep learning with LSTM based distributed data mining model for energy efficient wireless sensor networks. Physical Communication 40, 101097.

Mostafaei, H., Montieri, A., Persico, V., Pescape, A., 2017. A sleep scheduling approach based on learning automata for WSN partialcoverage. Journal of Network and Computer Applications 80, 67–78.

Multivariate and Univariate Time Series Prediction [WWW Document], 2024. URL https://in.mathworks.com/matlabcentral/fileexchange/69506-multivariate-and-univariate-time-series-prediction (accessed 6.29.24).

Narayan, V., Daniel, A.K., Chaturvedi, P., 2023. E-FEERP: Enhanced Fuzzy Based Energy Efficient Routing Protocol for Wireless Sensor Network. Wireless Pers Commun 131, 371–398. https://doi.org/10.1007/s11277-023-10434-z

Nilsaz Dezfuli, N., Barati, H., 2019. Distributed energy efficient algorithm for ensuring coverage of wireless sensor networks. IET Communications 13, 578–584. https://doi.org/10.1049/iet-com.2018.5329

Ovabor, K., Atkison, T., 2024. Exploring Firefly and Greywolf Algorithms for Multi-objective Optimization in Wireless Sensor Networks, in: Proceedings of the 2024 ACM Southeast Conference on ZZZ. Presented at the ACM SE '24: 2024 ACM Southeast Conference, ACM, Marietta GA USA, pp. 241–246. https://doi.org/10.1145/3603287.3651185

Pandiyaraju, V., Ganapathy, S., Mohith, N., Kannan, A., 2023a. An optimal energy utilization model for precision agriculture in WSNs using multi-objective clustering and deep learning. Journal of King Saud University-Computer and Information Sciences 35, 101803.

Pandiyaraju, V., Ganapathy, S., Mohith, N., Kannan, A., 2023b. An optimal energy utilization model for precision agriculture in WSNs using multi-objective clustering and deep learning. Journal of King Saud University-Computer and Information Sciences 35, 101803.

Patra, C., Guha Roy, A., Chattopadhyay, S., Bhaumik, P., 2010. Designing Energy-Efficient Topologies for Wireless Sensor Network: Neural Approach. International Journal of Distributed Sensor Networks 6, 216716. https://doi.org/10.1155/2010/216716

Purushothaman, K.E., Nagarajan, V., 2021. Multiobjective optimization based on self-organizing Particle Swarm Optimization algorithm for massive MIMO 5G wireless network. International Journal of Communication Systems 34, e4725.

Qihua, W., Ge, G., Lijie, C., Xufeng, X., 2015. Scheduling strategy for Hidden Markov Model in wireless sensor network, in: 2015 34th Chinese Control Conference (CCC). IEEE, pp. 7806–7810.

Rausand, M., 2014. Reliability of safety-critical systems: theory and applications. John Wiley & Sons.

Rudati, P.S., Feriyonika, F., Sudarsa, Y., Monda, H.T., 2023. OPTIMIZATION OF ENERGY CONSUMPTION IN SENSOR NODE BASED ON RECEIVED SIGNAL STRENGTH INDICATOR (RSSI) AND SLEEP AWAKE METHOD. ASEAN Engineering Journal 13, 153–158.

Sailhan, F., Delot, T., Pathak, A., Puech, A., Roy, M., 2009. Dependable wireless sensor networks, in: 3th Workshop Gestion Des Données Dans Les Systèmes d'Information Pervasifs (GEDSIP) in Cunjunction with INFORSID. pp. 1–16.

Salmi, S., Oughdir, L., 2022. Cnn-lstm based approach for dos attacks detection in wireless sensor networks. International Journal of Advanced Computer Science and Applications 13.

Saravanakumar, R., Mohankumar, N., Raja, J., 2013. Proficient Node Scheduling Protocol for Homogeneous and Heterogeneous Wireless Sensor Networks. International Journal of Distributed Sensor Networks 9, 826482. https://doi.org/10.1155/2013/826482

Saravanakumar, R., Susila, S.G., Raja, J., 2010. An energy efficient cluster based node scheduling protocol for wireless sensor networks, in: 2010 10th IEEE International Conference on Solid-State and Integrated Circuit Technology. IEEE, pp. 2053–2057.

Sendra, S., Parra, L., Lloret, J., Khan, S., 2015. Systems and algorithms for wireless sensor networks based on animal and natural behavior. International Journal of Distributed Sensor Networks 11, 625972.

Shagari, N.M., Idris, M.Y.I., Salleh, R.B., Ahmedy, I., Murtaza, G., Shehadeh, H.A., 2020. Heterogeneous energy and traffic aware sleep-awake cluster-based routing protocol for wireless sensor network. IEEE Access 8, 12232–12252.

Sharma, A., Kansal, A., 2023. Advanced ANN Based Secured Energy Efficient Routing Protocol in WSN. Wireless Pers Commun 132, 2645–2666. https://doi.org/10.1007/s11277-023-10737-1

Sharma, N.K., Tiwari, P.K., Sood, Y.R., 2011. Review of artificial intelligence techniques application to dissolved gas analysis on power transformer. International Journal of Computer and Electrical Engineering 3, 577–582.

Sharma, S., Sethi, D., Bhattacharya, P.P., 2015. Artificial neural network based cluster head selection in wireless sensor network. International Journal of Computer Applications 119.

Shu, T., Chen, J., Bhargava, V.K., de Silva, C.W., 2019. An energy-efficient dual prediction scheme using LMS filter and LSTM in wireless sensor networks for environment monitoring. IEEE Internet of Things Journal 6, 6736–6747.

Singh, A., Sharma, S., Singh, J., 2021. Nature-inspired algorithms for wireless sensor networks: A comprehensive survey. Computer Science Review 39, 100342.

Sohraby, K., Minoli, D., Znati, T., 2007a. Wireless Sensor Networks: Technology, Protocols, and Applications. John Wiley & Sons.

Sohraby, K., Minoli, D., Znati, T., 2007b. Wireless sensor networks: technology, protocols, and applications. John wiley & sons.

Srivastava, R., Hasan, M., 2018. Selective Sleep-awake Scheduling in WSN-Cloud Integration, in: 2018 5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON). IEEE, pp. 1–6.

Stehlık, M., 2011. Comparison of simulators for wireless sensor networks. Ph. D. dissertation, Masaryk University.

Taleb, A.A., 2021. Sink mobility model for wireless sensor networks using Kohonen self-organizing map. International Journal of Communication Networks and Information Security 13, 62–67.

Talei, H., Benhaddou, D., Gamarra, C., Benhaddou, M., Essaaidi, M., 2023. Identifying Energy Inefficiencies Using Self-Organizing Maps: Case of A Highly Efficient Certified Office Building. Applied Sciences 13, 1666.

Tate, J., Bate, I., 2011. LIPS: A Protocol Suite for Homeostatic Sensornet Management, in: Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference On. IEEE, pp. 263–272.

Testa, A., 2013. Dependability Assessment of Wireless Sensor Networks with Formal Methods.

Thomas, D., Shankaran, R., Sheng, Q.Z., Orgun, M.A., Hitchens, M., Masud, M., Ni, W., Mukhopadhyay, S.C., Piran, M.J., 2020. QoS-aware energy management and node scheduling schemes for sensor network-based surveillance applications. IEEE Access 9, 3065–3096.

Tian, D., Georganas, N.D., 2002. A Coverage-preserving Node Scheduling Scheme for Large Wireless Sensor Networks, in: Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications, WSNA '02. ACM, New York, NY, USA, pp. 32–41. https://doi.org/10.1145/570738.570744

Timo, D.H., others, 2005. Embedded Computer Systems: Architectures, Modeling, and Simulation: 5th International Workshop, SAMOS 2005, Samos, Greece, July 18-20, Proceedings. Springer Science & Business Media.

Vashisht, N., 2023. Achievement of Good Energy Efficiency in WSN using Sleep/Awake Technology, in: 2023 3rd Asian Conference on Innovation in Technology (ASIANCON). IEEE, pp. 1–7.

Vidhya, S., 2023. A survey on artificial neural network based energy-efficient and robust routing scheme for pollution monitoring in WSNs, in: Journal of Physics: Conference Series. IOP Publishing, p. 012048.

Viera, M.A.M., Viera, L.F.M., Ruiz, L.B., Loureiro, A.A., Fernandes, A.O., Nogueira, J.M.S., 2003. Scheduling nodes in wireless sensor networks: A Voronoi approach, in: Local Computer Networks, 2003. LCN'03. Proceedings. 28th Annual IEEE International Conference On. IEEE, pp. 423–429.

Wan, R., Xiong, N., Loc, N.T., 2018. An energy-efficient sleep scheduling mechanism with similarity measure for wireless sensor networks. Hum. Cent. Comput. Inf. Sci. 8, 18. https://doi.org/10.1186/s13673-018-0141-x

Wang, L., Wei, R., Lin, Y., Wang, B., 2010. A clique base node scheduling method for wireless sensor networks. Journal of Network and Computer Applications 33, 383–396.

Wang, X., Xing, G., Zhang, Y., Lu, C., Pless, R., Gill, C., 2003. Integrated coverage and connectivity configuration in wireless sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems. pp. 28–39.

Widiasari, I.R., Nugoho, L.E., Efendi, R., 2018. Context-based hydrology time series data for a flood prediction model using LSTM, in: 2018 5th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE). IEEE, pp. 385–390.

Wu, K., Gao, Y., Li, F., Xiao, Y., 2005. Lightweight deployment-aware scheduling for wireless sensor networks. Mobile networks and applications 10, 837–852.

Xianglin, Q., Haibing, Z., Yingying, Z., 2012. Research on wireless sensor networks clustering routing algorithm based on energy balance, in: Proceedings of 2012 International Conference on Measurement, Information and Control. IEEE, pp. 387–391.

Xu, Y., Heidemann, J., Estrin, D., 2001. Geography-informed energy conservation for ad hoc routing, in: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking. ACM, pp. 70–84.

Yang, J., Xia, Y., 2024. Coverage and Routing Optimization of Wireless Sensor Networks Using Improved Cuckoo Algorithm. IEEE Access.

Yang, O., 2011. Sleeping strategies for wireless sensor networks. University of Rochester.

Ye, F., Zhong, G., Cheng, J., Lu, S., Zhang, L., 2003. PEAS: A robust energy conserving protocol for long-lived sensor networks, in: Distributed Computing Systems, 2003. Proceedings. 23rd International Conference On. IEEE, pp. 28–37.

Zhang, T., Xue, C., Wang, J., Yun, Z., Lin, N., Han, S., 2024. A Survey on Industrial Internet of Things (IIoT) Testbeds for Connectivity Research. https://doi.org/10.48550/arXiv.2404.17485

Zroug, S., Remadna, I., Kahloul, L., Terrissa, S.L., Benharzallah, S., 2023. Towards performance evaluation prediction in WSNs using artificial neural network multi-perceptron. Cluster Comput 26, 1405–1423. https://doi.org/10.1007/s10586-022-03753-6

Appendices

Appendix A - Experiment 1: WSN without A Scheduling Algorithm code

```
% Clean memory and command window
clear,clc,close all
```

Parameters

```
m = 1;                          % Number of different types of 
sensors
rates = 80;                     % Packet size for normal node per 
round (bits) for each type
R = 15;                         % Range for cluster for each type

N = 150;              % Number of nodes
W = 100;              % length of the network
L = 100;              % width of the network
Ei = 3;               % Initial energy of each node (joules)
CHpl = 3000;          % Packet size for cluster head per round (bits)
p = 5/100;            % desired percentage of cluster heads
num_rounds = 2000;    % Max Number of simulated rounds
Tsetup = 4;           % average Time in seconds taken in setup phase
Tss = 10;             % average Time in seconds taken in steady state 
phase
Etrans = 1.0000e-09;  % Energy for transmitting one bit
Erec = 1.0000e-09;    % Energy for receiving one bit
Eagg = 1.0000e-09;    % Data aggregation energy
Efs = 5.00e-8;        % Energy of free space model amplifier
Marg = 15;            % Distance from the sink to sensors area
Rsense = 7;           % radius for sensing area around each node

Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33 
60.74];

pMin = 10^-4;         % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please 
check parameters')
end
```

Building the WSN

1st row: Clustering indexing 2nd: x-position, 3rd: y-position

```
net = [zeros(1,N);rand([1,N])*W;rand([1,N])*L;randi(m,[1,N])];
tmp = sqrt(net(2,:).^2 + net(3,:).^2); [~,I] = sort(tmp); net = 
net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end

% Position of sink
SX = -Marg; SY = L/2;
```

```
% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];
```

Start communication simulation

```
% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
end

% Preallocation for energy calculations
E = Ei*ones(1,N);           % Energy left in each node
EH = zeros(1,num_rounds);
% Preallocation for dead nodes calculations
Ecrit = 0;                  % Critical energy left in node to call it
alive
Ncrit = fix((90/100)*N);    % Critical number for dead nodes to stop
simulation
Dead = false(1,N);          % Status of all nodes 0:Alive 1:Dead
DeadH = zeros(1,num_rounds);
Cover = zeros(1,num_rounds);
% Preallocation for Bits sent calculations
BitsH = zeros(1,num_rounds);
BitsLost = zeros(1,num_rounds);
AED = zeros(1,num_rounds);
BitsNode = zeros(1,N);
TotSentCH = 0; TotRecCH = 0;
TotSentSink = 0; TotRecSink = 0;
figure('Position',[68,25,1347,875]);

% Simulating for each round
for r=1:num_rounds % iterating on each round

    % Updating alive status to all nodes
    Dead(E<=Ecrit) = true; E(Dead) = 0; DeadH(r)=sum(double(Dead));

    %%%% Choosing Clusters heads %%%%
    [CH,net,Num_itr] = UpdateCH(ranges,Dead,p,pMin,E,Ei,net,cost);

    %%%%%%%% Energy calculations %%%%%%%
    EH(r) = sum(E); %get all energy left in all nodes
```

first : Clustering stage %%%

```
    E = E - ((NonCHpl*(Etrans+Eagg)).*Num_itr);
```

second : Steady state stage %%%

Cluster heads energy calculations

```
    numClust = length(find(CH));
    D = sqrt((net(2,CH) - SX).^2 + (net(3,CH) - SY).^2);
    E(CH) = E(CH) - (((Etrans+Eagg)*CHpl)+(Efs*CHpl*(D.^
2))+(mean(NonCHpl)*Erec*round(N/numClust)));
    % Rest of nodes energy calculations
    rest = N-numClust-sum(double(Dead));
    mD = zeros(1,rest); tmp = net(2:3,~CH&~Dead);
    for i=1:rest, mD(i) = fun(tmp(1,i),tmp(2,i),net,CH,SX,SY); end
    E(~CH&~Dead) = E(~CH&~Dead) - ((mean(NonCHpl)*Etrans) +
(Efs*CHpl*(mD.^2)) + ((Erec+Eagg)*CHpl));
```

sent bits considering bucket loss %%%%

```
    for i=1:N
        if CH(i)==true
            Di = sqrt(((net(2,i)-SX).^2) + ((net(3,i)-SY).^2));
            delay = (3.3e-9)*Di - (1.7e-8);
            AED(r) = AED(r) + delay;
            tmp = P_loss(Di,Ta);
            BitsNode(i) = round((1-tmp)*CHpl);
            BitsLost(r) = BitsLost(r) + round(tmp*CHpl);
            TotRecSink = TotRecSink + BitsNode(i);
        else
            idx2 = find(((net(1,:)==net(1,i)) & CH) & ~Dead);
            Di = sqrt(((net(2,i)-net(2,idx2)).^2) + ((net(3,i)-
net(3,idx2)).^2));
            if isempty(Di)
                BitsNode(i) = 0;
            else
                tmp = P_loss(Di,Ta);
                BitsNode(i) = round((1-tmp)*NonCHpl(i));
                BitsLost(r) = BitsLost(r) + round(tmp*NonCHpl(i));
                TotRecCH = TotRecCH + BitsNode(i);
            end
        end
    end
    if r==1
        BitsH(r) = sum(BitsNode);
    else
        BitsH(r) = BitsH(r-1) + sum(BitsNode);
    end
    TotSentCH = TotSentCH + sum(NonCHpl(~CH&~Dead));
    TotSentSink = TotSentSink + numClust*CHpl;
    BitsLost(r) = BitsLost(r)/(TotSentCH+TotSentSink);
    AED(r) = AED(r)/sum(CH);

    % Coverability calculation
    warning off
    netAlive = net(:,~Dead);
    [Xcov1,Ycov1] = Make_cir(netAlive(2,1),netAlive(3,1),Rsense);
    for i=2:size(netAlive,2)
        [X,Y] = Make_cir(netAlive(2,i),netAlive(3,i),Rsense);
        [Xcov1, Ycov1] = polybool('union', Xcov1, Ycov1, X, Y);
    end
    [Xuncov, Yuncov] = polybool('subtraction', pgonx, pgony, Xcov1,
Ycov1);
    idxnan = find(isnan(Xuncov)); idxnan =
[0,idxnan,length(Xuncov)+1];
    A = 0;
    for i=1:length(idxnan)-1
```

```
        tmpx = Xuncov(idxnan(i)+1:idxnan(i+1)-1);
        tmpy = Yuncov(idxnan(i)+1:idxnan(i+1)-1);
        A = A + polyarea(tmpx, tmpy);
    end
    Cover(r) = 1 - (A/(W*L));
    warning on

    %%%% Showing updated net %%%%
    net = DrawNet(net,N,CH,Dead,SX,SY,Xcov1,Ycov1);
    title(['CH : Red  ----  Dead : Empty  ---  round
(',num2str(r),')']);
    drawnow
    if sum(Dead)>=Ncrit,break;end % Stop simulation when 5% or less
is alive
end
close all
figure('Position',[664,140,1170,775]);
net = DrawNet(net,N,CH,Dead,SX,SY,Xcov1,Ycov1);
title(['CH : Red  ----  Dead : Empty  ---  round (',num2str(r),')']);
T = (Tsetup+Tss)*(0:r-1);
EH = EH(1:r); EHdis = EH;
DeadH = DeadH(1:r); AliveH = N-DeadH;
BitsH = BitsH(1:r);
BitsLost = BitsLost(1:r);
AED = AED(1:r);
Cover = Cover(1:r);

PDR_CH = TotRecCH/TotSentCH;
PDR_Sink = TotRecSink/TotSentSink;

disp('Analysis:')
disp('=========================')
disp(['Packet delivery ratio to the cluster heads =
',num2str(100*PDR_CH),'%'])
disp(['Packet delivery ratio to sink = ',num2str(100*PDR_Sink),'%'])
disp('')
```

Plotting analysis of network preformance

```
figure('Position',[131 59 792 613]);
plot(T,EHdis,'-x');
xlabel('Time (s)'); ylabel('Energy (j)')
title('Residual energy')

figure('Position',[298 66 792 613]);
plot(N-AliveH,'-x');
xlabel('Round'); ylabel('No of dead Sensors nodes')
title('Life time of sensor nodes')

figure('Position',[511 61 792 613]);
plot(T,BitsH,'-x');
xlabel('Time (s)'); ylabel('Overall Throughput (no of packets)')
title(['Throughput (' num2str(N) ' Sensor nodes)'])

figure('Position',[611 61 792 613]);
plot(1-BitsLost,'-x')
xlabel('Round');
title('Connectivity')
```

```
figure('Position',[711 61 792 613]);
plot(Cover,'-x');
xlabel('Round'); ylabel('Coverability')
title('Ratio of coverd area to total area')
```

Appendix B - Experiment 2: WSN with RCS Scheduling Algorithm code

```
% Clean memory and command window
clear,clc,close all
```

Parameters

```
rates = 80;          % Packet size for normal node per round (bits)
for each type
R = 14;              % Max range for wireless transsmion for each
node

k = 4;               % Number of ON/OFF groups

N = 150;             % Number of nodes
W = 100;             % length of the network
L = 100;             % width of the network
Ei = 3;              % Initial energy of each node (joules)
CHpl = 3000;         % Packet size for cluster head per round (bits)
p = 10/100;          % desired percentage of cluster heads
num_rounds = 2000;   % Max Number of simulated rounds
Tsetup = 4;          % average Time in seconds taken in setup phase
Tss = 10;            % average Time in seconds taken in steady state
phase
Etrans = 1.0000e-09; % Energy for transmitting one bit
Erec = 1.0000e-09;   % Energy for receiving one bit
Eagg = 1.0000e-09;   % Data aggregation energy
Efs = 5.00e-8;       % Energy of free space model amplifier
Marg = 15;           % Distance from the sink to sensors area
Rsense = 7;          % radius for sensing area around each node

Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33
60.74];
m = 1;
pMin = 10^-4;        % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please
check parameters')
end
```

Building the WSN

1st row: Clustering indexing 2nd: x-position, 3rd: y-position

```
net = [zeros(1,N);rand([1,N])*W;rand([1,N])*L;randi(m,[1,N])];
tmp = sqrt(net(2,:).^2 + net(3,:).^2); [~,I] = sort(tmp); net =
net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
```

```
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end

Groups = randi(k,[1,N]); % ON/OFF grouping

% Position of sink
SX = -Marg; SY = L/2;

% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];

% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
end
```

Start communication simulation

```
% first run no scheduling
USE_schedule = 0;
simulate_WSN
```

second run using scheduling

```
USE_schedule = 1;
simulate_WSN

function [ON,net] = Scheduling(net,CH,r,k,Groups,USE,R)
if USE

    ON_group = mod(r,k)+1; % The choosen group to be ON this round
    ON_CH = find((Groups == ON_group) & CH); % the cluster heads that are
ON
    OFF_CH = find((Groups ~= ON_group) & CH); % the cluster heads that
are off

    % Extra-On Rule
    X = net(2,:); Y = net(3,:);
    for i=1:length(ON_CH)
        sx = X(ON_CH(i+1:end)); ix = X(ON_CH(i));
        sy = Y(ON_CH(i+1:end)); iy = Y(ON_CH(i));
        d = sqrt((sx - ix).^2 + (sy - iy).^2);
        d = min(d);
        if d > R
            sx = X(OFF_CH);
            sy = Y(OFF_CH);
            d = sqrt((sx - ix).^2 + (sy - iy).^2);
            [~,d] = min(d);
            % rempve it from off cluster heads and put it in ON ones
            if ~isempty(d)
            ON_CH(end+1) = d;
            OFF_CH(d) = [];
            end
        end
    end

    % Turning ON the cluster heads and their children
    ON = false(1,size(net,2));
    for i=1:length(ON_CH)
        c = net(1,ON_CH(i));
        ON(net(1,:)==c) = true;
    end

    % remove OFF nodes from clustering
    net(1,~ON) = 0;


else
    ON = true(1,size(net,2));
end
```

Appendix C - Replication of Randomised Coverage Based Scheduling Algorithm RCS code

```
% Clean memory and command window
clear,clc,close all

% Load the trained neural network
addpath('codes') ; % load codes library
load('Nets.mat')
nnet = Nets.Net1;
```

Parameters

```
k = 3;                  % Number of ON/OFF groups
N = 100;                % Number of nodes
Rsense = 1;             % radius for sensing area around each node
W = 20;                 % length of the network
L = 20;                 % width of the network

Ei = 0.25;              % Initial energy of each node (joules)
CHpl = 3000;            % Packet size for cluster head per round (bits)
p = 5/100;              % desired percentage of cluster heads
num_rounds = 2000;      % Max Number of simulated rounds
Tsetup = 4;             % average Time in seconds taken in setup phase
Tss = 10;              % average Time in seconds taken in steady state phase
Etrans = 1.0000e-09;    % Energy for transmitting one bit
Erec = 1.0000e-09;      % Energy for receiving one bit
Eagg = 1.0000e-09;      % Data aggregation energy
Efs = 5.00e-8;          % Energy of free space model amplifier
Marg = 5;               % Distance from the sink to sensors area
rates = 80;              % Packet size for normal node per round (bits) for
each type
R = 3.5;                  % Max range for wireless transition for each node
Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33 60.74];
m = 1;
```

```
pMin = 10^-4;         % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please check
parameters')
end
```

Building the WSN

```
% Position of sink
SX = -Marg; SY = L/2;

% 1st row: Clustering indexing
% 2nd: x-position, 3rd: y-position
net = [zeros(1,N) ; rand([1,N])*W ; rand([1,N])*L ; randi(m,[1,N])];
tmp = sqrt((net(2,:)-SX).^2 + (net(3,:)-SY).^2);
[~,I] = sort(tmp,'descend');
net = net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end

Groups = randi(k,[1,N]); % ON/OFF grouping
ON_pr = [];
% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];

% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
    % special case of isolated node
    if isnan(cost(i)),cost(i)=max(Dist);end
end
```

Use Neural network to predict Extra_on nodes

```
Xnet = ((net(2,:) - SX)./R)';
Ynet = ((net(3,:) - SY)./R)';
ON_pr = predictNN([Xnet,Ynet], nnet); % the prediction
```

Start communication simulation

```
SHOWPLOTS = 1;

% first run no scheduling
USE_schedule = 0;
simulate_WSN
```

second run using scheduling

```
USE_schedule = 1;
simulate_WSN
```

```
function [ON,net,ExtraOn] = Scheduling(net,CH,r,k,Groups,USE,R,D)

ExtraOn = false(1,size(net,2));

if USE

    ON_group = mod(r,k)+1; % The chosen group to be ON this round
    ON_CH = find((Groups == ON_group) & CH & ~D); % the cluster heads that
are ON
    OFF_CH = find((Groups ~= ON_group) & CH & ~D); % the cluster heads
that are off

    % Extra-On Rule
    X = net(2,:); Y = net(3,:);
    len = length(ON_CH);
    for i=1:len-1
        sx = X(ON_CH(i+1:len)); ix = X(ON_CH(i));
        sy = Y(ON_CH(i+1:len)); iy = Y(ON_CH(i));
        d = sqrt((sx - ix).^2 + (sy - iy).^2);
        d = min(d);
        if d > R
            sx = X(OFF_CH);
            sy = Y(OFF_CH);
            d = sqrt((sx - ix).^2 + (sy - iy).^2);
            [~,d] = min(d);
            % remove it from off cluster heads and put it in ON ones
            if ~isempty(d)
            ON_CH(end+1) = OFF_CH(d);

            ExtraOn(OFF_CH(d)) = true;


            OFF_CH(d) = [];
            end
        end
    end

    % Turning ON the cluster heads and their children
    ON = false(1,size(net,2));
    for i=1:length(ON_CH)
        c = net(1,ON_CH(i));
        ON(net(1,:)==c) = true;
    end

    % remove OFF nodes from clustering
    net(1,~ON) = 0;


else
    ON = true(1,size(net,2));
end
```

Appendix D - A Perceptron-Based Neural-Network Analyser to Enhance the Scheduling Performance in WSN code

```
% This code makes the data samples and add them to the folder named
% Database .... I already created a lot
clear,clc,close all
% number of files already in Database folder
Numfiles = dir('Database'); Numfiles = length(Numfiles)-2;
% number of new sample files you want to create
NumToCreate = 1;

% Parameters
rates = 80;           % Packet size for normal node per round (bits) for
each type
R = 15;               % Max range for wireless transition for each node

k = 2;                % Number of ON/OFF groups

N = 120;              % Number of nodes
W = 100;              % length of the network
L = 100;              % width of the network
Ei = 3;               % Initial energy of each node (joules)
CHpl = 3000;          % Packet size for cluster head per round (bits)
p = 5/100;           % desired percentage of cluster heads
num_rounds = 2000;    % Max Number of simulated rounds
Tsetup = 4;           % average Time in seconds taken in setup phase
Tss = 10;               % average Time in seconds taken in steady state
phase
Etrans = 1.0000e-09; % Energy for transmitting one bit
Erec = 1.0000e-09;    % Energy for receiving one bit
Eagg = 1.0000e-09;    % Data aggregation energy
Efs = 5.00e-8;        % Energy of free space model amplifier
Marg = 15;            % Distance from the sink to sensors area
Rsense = 5;           % radius for sensing area around each node

Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33 60.74];
m = 1;
% Position of sink
SX = -Marg; SY = L/2;
```

Start creating samples

```
WB = waitbar(0,'Creating data samples .... please wait');
for KK=1:NumToCreate

pMin = 10^-4;         % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please check
parameters')
end

% Building the WSN
% 1st row: Clustering indexing
% 2nd: x-position, 3rd: y-position
net = [zeros(1,N) ; rand([1,N])*W ; rand([1,N])*L ; randi(m,[1,N])];
tmp = sqrt((net(2,:)-SX).^2 + (net(3,:)-SY).^2);
[~,I] = sort(tmp,'descend');
net = net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
```

```
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end

Groups = randi(k,[1,N]); % ON/OFF grouping

% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];

% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
    % special case of isolated node
    if isnan(cost(i)),cost(i)=max(Dist);end
end

% run using scheduling
USE_schedule = 1;
simulate_WSN

% Save this sample to a file in Database folder
save(['Database/test',num2str(Numfiles + KK),'.mat'],'Extra_ON','net')
waitbar(KK/NumToCreate,WB,'Creating data samples .... please wait')
end

close(WB)
```

```
% Once the data samples are created this code creates the database from
the
% Samples to be used in neural network training
% Clean memory and command window
clear,clc,close all

R  = 15;      % Max range for wireless transition for each node
SX = -15;     % X-coordinate of the base station (sink)
SY = 50;      % Y-coordinate of the base station (sink)

Files = dir('Database');
sample_data = [];
for i=3:length(Files)-1
    try
    load(fullfile('Database',Files(i).name))

    X = ((net(2,:) - SX)./R)'; % second row all columns
    Y = ((net(3,:) - SY)./R)';

    tmp = sum(Extra_ON)';

    class = zeros(size(tmp));
    class(tmp <  4) = 1;
    class(tmp >= 4) = 2;

    sample_data = [sample_data;[X,Y,class]];
    catch
    end
end

% Take the dataset parameters from user
n = fix(input('Enter the number of input variables: '));
m = fix(input('Enter the number of functions to fit: '));
train_ratio = input('Enter the ratio of data used for creating the model:
');
if n+m ~= size(sample_data,2)
    error('Your inputs are not consistent with data size');
end

% Divide data to train set and test set
totalSamples = size(sample_data,1);
[trainInd,~,testInd]      =      dividerand(totalSamples,train_ratio,0,1-
train_ratio);
Xtrain = sample_data(trainInd,1:n);
Xtest = sample_data(testInd,1:n);
Ytrain = sample_data(trainInd,n+1:end);
Ytest = sample_data(testInd,n+1:end);

clear sample_data testInd trainInd;
save('Data.mat');
disp('============================================================');
whos
```

```
% Once the data.mat file created to be used in NN training this is the
% code that does the training
% Initialization
clear; close all; clc; % clear memory of old values
addpath('codes') ; % load codes library

% Parameters
NumEpochs = 200;  % number of iterations
contFromLastTime = 0; % 0:false ,,, 1:true

%% 1) Import Data
load('Data.mat');
numOut = size(Ytrain,2);
Ytest_orig = Ytest; % save original test output values
Nets = struct(); % Initialize Nets with empty struct

%% 2) Build NN architecture
options = {'lambda', 0.1,...
    'maxIter', NumEpochs ,...
    'hiddenLayers', [40 30 20],...
    'validPercent', 30,...
    'doNormalize', 1};

%% 3) Combine all in one system for training
% Approximate values of output to discrete classes
for i=1:numOut
    disp(' '),disp(['Output number ',num2str(i),' :'])
    disp('=======================================')

    if ~contFromLastTime
        tmpNet = learnNN(Xtrain,Ytrain,contFromLastTime,options);
    else
        tmpNet = learnNN(Xtrain,Ytrain,i,options);
    end
    Nets = setfield(Nets,['Net',num2str(i)],tmpNet);

    % Testing to get accuracy
    p = predictNN(Xtest, tmpNet); % the prediction
    correctIDX = (p-Ytest)==0;
    Accuracy_classification = 100*sum(correctIDX)/size(Ytest,1);
    disp(['Accuracy             of            classification           =
',num2str(Accuracy_classification)])

end

% Saving trained Model
save('Nets','Nets');
```

Appendix E - The Hidden Markov Model (HMM) Node Scheduling Algorithm code

The code of the described HMM function is borrowed from the below reference:

[“Hidden Markov model states and emissions - MATLAB hmmgenerate - MathWorks India,” n.d.]

```
% The function performs the state probabilities considering an input as
described below when a particular input is taken to achieve its full state

function [tr]= RegenerateMarkov(L)

T1 = [0.95,0.05; 0.90,0.90];
T2 = [1/6 1/6 1/6 1/6 1/6 1/6;
   1/10 1/10 1/10 1/10 1/10 1/2];


seq = zeros(1,L);
states = zeros(1,L);

% T1 Should be a square matrix

numStates = size(T1,1);
checkTr = size(T1,2);
if checkTr ~= numStates
    error(message('stats:RegenerateMarkov:BadTransitions'));
end

% The number of rows of e must be same as number of states

checkE = size(T2,1);
if checkE ~= numStates
    error(message('stats:RegenerateMarkov:InputSizeMismatch'));
end

numEmissions = size(T2,2);

customSymbols = false;
customStatenames = false;

if nargin > 3
    okargs = {'symbols','statenames'};
    [symbols,statenames] = ...
        internal.stats.parseArgs(okargs, {[] []}, varargin{:});

    if ~isempty(symbols)
        numSymbolNames = numel(symbols);
        if ~isvector(symbols) || numSymbolNames ~= numEmissions
            error(message('stats:RegenerateMarkov:illegalSymbols'));
        end
        customSymbols = true;
    end
    if ~isempty(statenames)
        numStateNames = length(statenames);
        if numStateNames ~= numStates

error(message('statistics:RegenerateMarkov:illegalStateNames'));
        end
        customStatenames = true;
    end
end

% create two random sequences, one for state changes, one for emission
statechange = rand(1,L);
randvals = rand(1,L);

% calculate cumulative probabilities
trc = cumsum(T1,2);
ec = cumsum(T2,2);

% normalize these just in case they don't sum to 1.
trc = trc./repmat(trc(:,end),1,numStates);
ec = ec./repmat(ec(:,end),1,numEmissions);

% Assume that we start in state 1.
currentstate = 1;

% main loop
for count = 1:L
    % calculate state transition
    stateVal = statechange(count);
    state = 1;
    for innerState = numStates-1:-1:1
        if stateVal > trc(currentstate,innerState)
            state = innerState + 1;
            break;
        end
    end
    % calculate emission
    val = randvals(count);
    emit = 1;
    for inner = numEmissions-1:-1:1
        if val  > ec(state,inner)
            emit = inner + 1;
            break
        end
    end
    % add values and states to output
    seq(count) = emit;
    states(count) = state;
    currentstate = state;
end

% deal with names/symbols
if customSymbols
    seq = reshape(symbols(seq),1,L);
end
if customStatenames
    states = reshape(statenames(states),1,L);
end
if nargin > 0
```

```
    if isstring(seq)
        seq = cellstr(seq);
    end
end

if nargin > 1
    if isstring(states)
        states = cellstr(states);
    end
end

if nargin > 2
    [varargin{:}] = convertStringsToChars(varargin{:});
end

pseudoEcounts = false;
pseudoTRcounts = false;
tr = [];
E = [];
seqLen = length(seq);
if length(states) ~= seqLen
    error(message('InputMismatch'));
end
uniqueSymbols = unique(seq);
uniqueStates = unique(states);

if isempty(uniqueSymbols)
    warning(message('EmptySequence'));
    return
end
if isempty(uniqueStates)
    warning(message('EmptyState'));
    return
end

% If sequence(seq) is numeric, assume that the number of symbols is the
largest
% number in seq Otherwise, assume it's the number of unique symbols in
% seq Use the same logic for the number of states.
if isnumeric(seq)
    numSymbols = max(uniqueSymbols);
else
    numSymbols = length(uniqueSymbols);
end
if isnumeric(states)
    numStates = max(uniqueStates);
else
    numStates = length(uniqueStates);
end

if nargin > 2
    okargs                                                              =
{'symbols','statenames','pseudoemissions','pseudotransitions'};
    [symbols,statenames,pseudoE,pseudoTR,setflag] = ...
        internal.stats.parseArgs(okargs, {[] [] [] []}, varargin{:});

    if setflag.symbols
```

```
        numSymbols = numel(symbols);
        if length(symbols) ~= numSymbols
            error(message('BadSymbols'));
        end
        [~, seq]  = ismember(seq,symbols);
        if any(seq == 0)
            error(message('SymbolNotInSymbolNames'));
        end
    end
    if setflag.statenames
        numStates = numel(statenames);
        if length(statenames) ~= numStates
            error(message('BadStateNames'));
        end
        [~, states]  = ismember(states,statenames);
        if any(states == 0)
            error(message('StateNotInStateNames'));
        end
    end
    if setflag.pseudoemissions
        [rows, cols] = size(pseudoE);
        if  rows < numStates
            error(message('StatesPseudoMismatch'));
        end
        if  cols < numSymbols
            error(message('SeqPseudoMismatch'));
        end
        numStates = rows;
        numSymbols = cols;
        pseudoEcounts = true;
    end
    if setflag.pseudotransitions
        [rows, cols] = size(pseudoTR);
        if rows ~= cols
            error(message('BadTransitions'));
        end
        if  rows < numStates
            error(message('StatesPseudoMismatch'));
        end
        numStates = rows;
        pseudoTRcounts = true;
    end
end


tr = zeros(numStates);
E = zeros(numStates, numSymbols);
% count up the transitions from the state path
for count = 1:seqLen-1
    tr(states(count),states(count+1)) = tr(states(count),states(count+1))
+ 1;
end

if pseudoTRcounts
    tr = tr + pseudoTR;
end
```

```
trRowSum = sum(tr,2);


% If we don't have any values then report zeros instead of NaNs (Error
correction in case of Not a Number)
.
trRowSum(trRowSum == 0) = -inf;


% Normalize to give frequency estimate.
tr = tr./repmat(trRowSum,1,numStates);
```

Appendix F - The Bio-inspired BAT Node Scheduling Algorithm Code

```
% Clean memory and command window
clear;
clc;
close all;

% Load the trained neural network
addpath('codes') ; % load codes library
% load('Nets.mat')
%
% nnet = Nets.Net1;
N = 100;                 % Number of nodes
 bat = test_bat(N);
 bat = bat';
```

Parameters

```
 k = ceil(bat(1,49));                 % Number of ON/OFF groups

%k=3;


W = 20;                  % length of the network
L = 20;                  % width of the network

Ei = 0.25;               % Initial energy of each node (joules)
CHpl = 3000;             % Packet size for cluster head per round (bits)
p = 5/100;               % desired percentage of cluster heads
num_rounds = 2000;       % Max Number of simulated rounds
Tsetup = 4;              % average Time in seconds taken in setup phase
Tss = 10;                  % average Time in seconds taken in steady state
phase
Etrans = 1.0000e-09;     % Energy for transmitting one bit
Erec = 1.0000e-09;       % Energy for receiving one bit
Eagg = 1.0000e-09;       % Data aggregation energy
Efs = 5.00e-8;           % Energy of free space model amplifier
Marg = 5;                % Distance from the sink to sensors area
rates = 80;              % Packet size for normal node per round (bits) for
each type
```

```
R = 3.5;                  % Max range for wireless transmition for each
node
Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33 60.74];
m = 1;
pMin = 10^-4;         % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please check
parameters')
end
```

Building the WSN

```
% Position of sink
SX = -Marg; SY = L/2;

% 1st row: Clustering indexing
% 2nd: x-position, 3rd: y-position

net = [zeros(1,N) ; rand([1,N])*W ; rand([1,N])*L ; randi(m,[1,N])];
% net = [zeros(1,N) ; bat(1,:)*W ; bat(2,:)*L ; randi(m,[1,N])];
tmp = sqrt((net(2,:)-SX).^2 + (net(3,:)-SY).^2);
[~,I] = sort(tmp,'descend');
net = net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end

Groups = randi(k,[1,N]); % ON/OFF grouping
ON_pr = [];
% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];

% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
    % special case of isolated node
    if isnan(cost(i)),cost(i)=max(Dist);end
end
```

Use Neural networkto predict Extra_on nodes

Xnet = ((net(2,:) - SX)./R)'; Ynet = ((net(3,:) - SY)./R)'; ON_pr = predictNN([Xnet,Ynet], nnet) % the prediction

```
%ON_pr = ceil(bat(1,50));
ON_pr =ceil(bat(2,:));
```

Start communication simulation

```
SHOWPLOTS = 1;
```

```
% first run no scheduling
USE_schedule = 0;
    Rsense = 1;           % radius for sensing area around each node
    simulate_WSN
%%=\]we3231% second run using scheduling
USE_schedule = 1;
    Rsense = ceil(bat(2,49));
    simulate_WSN



function P=test_bat(NPareto)
if nargin<1,
    NPareto;  % Number of points on the Pareto front
end
global w;
for k=1:NPareto,
    % Generate a weighting coefficient:w so that w1=w, w2=1-w, w1+w2=1.
    % ObserEations suggest that systematically monotonic weights are
    % better than random weights.
    w=k/NPareto;
    [best,fmin]=bat_algorithm;
    [o1,o2]=optimizer(best);
    P(k,:)=[o1,o2];
    % Output/display
  disp(['Weight: ',num2str(w)]);
  disp(['Best o1=',num2str(o1),'   o2=',num2str(o2)]);

end
% Display the Pareto front
% plot(P(:,1),P(:,2),'o');
% xlabel('o_1'); ylabel('o_2');
% The main part of the Bat Algorithm                       %
% Usage: bat_algorithm([20 0.25 0.5]);                     %
function [best,fmin,N_iter]=bat_algorithm(para)
% Default parameters
if nargin<1,  para=[10 0.25 0.5];  end
n=para(1);      % number of bats typically 10 to 150
A=para(2);      % population Density  (constant or decreasing)
r=para(3);      % Heart rate (constant or decreasing)

Pmin=0;         % heart beat freq minimum
Pmax=2;         % heart beat freq maximum
% Iteration parameters
```

In order to obtain better/more accurate results, N_iter

should be increased to N_iter=2000 or more if necessary.

```
N_iter=1000;       % Total number of function eEaluations
       % Dimension of the search variables
d=4;
% Initial arrays
P=zeros(n,1);   % Heart Rate
E=zeros(n,d);   %  Energy
% Initialize the population/solutions
```

```
for i=1:n,
  Sol(i,:)=randn(1,d);
  Fitness(i)=Fun(Sol(i,:));
end
% Find the current best
[fmin,I]=min(Fitness);
best=Sol(I,:);
% =======================================================  %
% Note: As this is a demo, here we did not implement the  %
% reduction of loudness and increase of emission rates.   %
% Interested readers can do some parametric studies       %
% and also implementation Earious changes of A and r etc  %
% =======================================================  %
% Start the iterations -- Bat Algorithm
for i_ter=1:N_iter,
        % Loop oEer all bats/solutions
        for i=1:n,
          P(i)=Pmin+(Pmin-Pmax)*rand;
          E(i,:)=E(i,:)+(Sol(i,:)-best)*P(i);
          S(i,:)=Sol(i,:)+E(i,:);
          % Heart rate
          if rand>r
              S(i,:)=best+0.01*randn(1,d);
          end
     % EEaluate new solutions
           Fnew=Fun(S(i,:));
     % If the solution improEes or not too loudness
           if (Fnew<=Fitness(i)) & (rand<A) ,
                Sol(i,:)=S(i,:);
                Fitness(i)=Fnew;
           end
          % Update the current best
          if Fnew<=fmin,
                best=S(i,:);
                fmin=Fnew;
          end
        end

end
% End of the main bat algorithm and output/display can be added here.
% Put your oectiEe functions here
function z=Fun(u)
global w;
[o1,o2]=optimizer(u);
z=o1*w+(1-w)*o2;
% Two oectiEes
function [o1,o2]=optimizer(u)
% In the simplest 1D case, f1=x^2, f2=(x-2)^2.
% In the d-dim case, the Pareto front extends from (0,4d) to (4d,0).
o1=sum(u.^2);
o2=sum((u-2).^2);
```

Appendix G - The Self-Organizing Feature Map (SOFM) Node Scheduling Algorithm code

```
% Clean memory and a command window
clear;
clc;
close all;

% Load the trained neural network
%addpath('codes') ; % load codes library
% load('Nets.mat')
%
% nnet = Nets.Net1;
 N = 100;                 % Number of nodes

C = readtable('coordinate.csv');
C= table2array(C);
S =readtable('som.csv'); %store som data
S= table2array(S);
S =S';
```

Parameters

```
     k = ceil(bat(1,39));                 % Number of ON/OFF groups
%  k = ceil(bat(49,1));
%k=3;


W = 20;                  % length of the network
L = 20;                  % width of the network

Ei = 0.25;             % Initial energy of each node (joules)
CHpl = 3000;           % Packet size for cluster head per round (bits)
p = 20/100;            % desired percentage of cluster heads
num_rounds = 200;      % Max Number of simulated rounds
Tsetup = 4;            % average Time in seconds taken in setup phase
Tss = 10;                % average Time in seconds taken in steady state
phase
Etrans = 1.0000e-09; % Energy for transmitting one bit
Erec = 1.0000e-09;   % Energy for receiving one bit
Eagg = 1.0000e-09;   % Data aggregation energy
Efs = 5.00e-8;       % Energy of free space model amplifier
Marg = 5;            % Distance from the sink to sensors area
rates = 80;         % Packet size for normal node per round (bits)  for
each type
R = 3.5;             % Max range for wireless transmition for each node
Ta = [10 0 2;20 0 4.459;30 0 3.75;40 1.38 7.14;50 2.09 20.37
      60 2.54 10.18;70 1.86 10;80 2.77 30;90 5.9 30.95;100 23.33 60.74];
m = 1;
pMin = 10^-4;          % Lowest possible CH_prop
if length(rates)~=m || length(R)~=m
    error('The length of rates and ranges is not equal to m please check
parameters')
end
```

Building the WSN

```
% Position of sink
SX = -Marg; SY = L/2;

% 1st row: Clustering indexing
% 2nd: x-position, 3rd: y-position

%net = [zeros(1,N) ; rand([1,N])*W ; rand([1,N])*L ; randi(m,[1,N])];
net = [zeros(1,N) ; C(:,1)'*W ; C(:,2)'*L ; randi(m,[1,N])];
% net = [zeros(1,N) ; bat(1,:)*W ; bat(2,:)*L ; randi(m,[1,N])];
tmp = sqrt((net(2,:)-SX).^2 + (net(3,:)-SY).^2);
[~,I] = sort(tmp,'descend');
net = net(:,I);
NonCHpl = zeros(1,N); ranges = zeros(1,N);
for i=1:m
    idx = net(end,:)==i;
    NonCHpl(idx) = rates(i);
    ranges(idx) = R(i);
end
%%03340291100
Groups = randi(k,[1,N]); % ON/OFF grouping
ON_pr = [];
% area to be sensed
pgonx = [0 0 W W];
pgony = [0 L L 0];

% calculating costs
cost = zeros(1,N);
for i=1:N
    Dist = sqrt(((net(2,:)-net(2,i)).^2) + ((net(3,:)-net(3,i)).^2));
    d = sqrt(((SX-net(2,i)).^2) + ((SY-net(3,i)).^2));
    Snbr = Dist <= ranges(i);
    cost(i) = sum(Dist(Snbr))/(sum(Snbr)-1)+d;
    % special case of isolated node
    if isnan(cost(i)),cost(i)=max(Dist);end
end
nnet = selforgmap([100 2]); %SOFM is created here
%(https://in.mathworks.com/help/deeplearning/ref/selforgmap.html)
nnet = train(nnet,C);
```

Xnet = ((C(:,1) - SX)./R)'; Ynet = ((C(:,2) - SY)./R)'; ON_pr = ([Xnet,Ynet], nnet); % the prediction

```
ON_pr=nnet(C);
```

Start communication simulation

```
SHOWPLOTS = 1;

% first run no scheduling
USE_schedule = 0;
    Rsense = 1;          % radius for sensing area around each node
    simulate_WSN
%%=\]we3231% second run using scheduling
USE_schedule = 1;
%     Rsense = ceil(bat(2,39));
%     %Rsense = ceil(bat(49,2));
     Rsense = 1;
     simulate_WSN
```

The LSTM code was modified to delete missing values, such as outlier values and noises and develop single best output amongst other possible values. For example, if the LSTM model is applied to WSN for 20 sec it can train on 20 secs data and predict upto next 20 sec data. The LSTM code was borrowed from the source below:

[“Multivariate and Univariate Time Series Prediction,” 2024]

```
%clc; clear; close all;
function data = TimeSeriesPrediction()
%% ---------------------------- init Variabels --------------------------
--

opt.Delays = [1 2 3 4 5 6 7 8 9 10 12 16 20];
opt.dataPreprocessMode     =  'Data  Standardization';  %  'None'  'Data
Standardization' 'Data Normalization'
opt.learningMethod      = 'LSTM';                  % 'MLP' 'LSTM'
opt.trPercentage        = 0.8;                    %  divide data into Test
and Train dataset

% ------------- BILSTM parameters
opt.NumOfHiddenLayers = 2;                           %  number of (bi)LSTM
layers

opt.NumOfUnitsInFirstlayer  = 100;                   %  number of (bi)LSTM
units in the first  layer
opt.NumOfUnitsInSecondlayer = 100;                   %  number of (bi)LSTM
units in the second layer
opt.NumOfUnitsInThirdlayer  = 75;                    %  number of (bi)LSTM
units in the third  layer
opt.NumOfUnitsInFourthlayer = 75;                    %  number of (bi)LSTM
units in the fourth  layer

opt.isUseBiLSTMLayer  = true;                    % if it is true the layer
turn to the Bidirectional-LSTM and if it is false it will turn the units
to the simple LSTM
opt.isUseDropoutLayer = true;                     % dropout layer avoid of
bieng overfit
opt.DropoutValue      = 0.5;

opt.maxEpochs      = 200;                              % maximum number of
training Epoch in bi-LSTM.
opt.miniBatchSize = 64;                            % minimum batch size in
bi-LSTM .
opt.executionEnvironment = 'cpu';                % 'cpu' 'gpu' 'auto'
opt.LR                  = 'adam';                % 'sgdm' 'rmsprop' 'adam'
opt.trainingProgress      = 'training-progress';   %  'training-progress'
'none'.

% MLP parameters
opt.NumOfFeedForwardLeyars = 2;

opt.NumOfNeuronsInFirstlayer  = 15;          %  number of neurons in the
first  layer
opt.NumOfNeuronsInSecondlayer = 15;          %  number of neurons in the
second layer
opt.NumOfNeuronsInThirdlayer  = 10;          %  number of neurons in the
third  layer
```

```
opt.trainFcn = 'trainbr';                 % 'trainlm' 'trainscg' 'traincgf'
'trainbr'
opt.maxItrations = 100;                      % maximum number of training
itration.
opt.showWindow              = true;          % display training window.
opt.showCommandLine         = true;          % display training process on
workspace.

opt.isSavePredictedData    = true;          %  save output prediction on an
excel file

%% --------------- load Data
data = loadData(opt);
if ~data.isDataRead
    return;
end
%% --------------- Train Network
[opt,data] = TrainData(opt,data);

%% --------------- Evaluate Data
[opt,data] = EvaluationData(opt,data);


%% ---------------------------- Local Functions --------------------------
--
function data = loadData(opt)
 [chosenfile,chosendirectory] = uigetfile({'*.xlsx';'*.csv'},...
                        'Select Excel time series Data sets','data.xlsx');
   filePath = [chosendirectory chosenfile];
    if filePath ~= 0
        data.DataFileName = chosenfile;
        data.CompleteData = readtable(filePath);
        if size(data.CompleteData,2)>1
             warning('Input data should be an excel file with only one
column!');
             disp('Operation Failed... '); pause(.9);
             disp('Reloading data. ');     pause(.9);
             data.x = [];
             data.isDataRead = false;
             return;
        end
        data.seriesdataHeder                                               =
data.CompleteData.Properties.VariableNames(1,:);
        data.seriesdata = table2array(data.CompleteData(:,:));
        disp('Input data successfully read.');
        data.isDataRead = true;
        data.seriesdata = PreInput(data.seriesdata);

%         figure('Name','InputData','NumberTitle','off');
%         plot(data.seriesdata);
%            title({['Mean  =  '  num2str(mean(data.seriesdata))  ',  STD  =  '
num2str(std(data.seriesdata)) ];});
        if strcmpi(opt.dataPreprocessMode,'None')
            data.x = data.seriesdata;
        elseif strcmpi(opt.dataPreprocessMode,'Data Normalization')
            data.x = DataNormalization(data.seriesdata);
%             figure('Name','NormilizedInputData','NumberTitle','off');
%             plot(data.x); grid minor;
%                   title({['Mean  =  '  num2str(mean(data.x))  ',  STD  =  '
num2str(std(data.x)) ];});
```

```
        elseif strcmpi(opt.dataPreprocessMode,'Data Standardization')
            data.x = DataStandardization(data.seriesdata);
%              figure('Name','NormilizedInputData','NumberTitle','off');
%              plot(data.x); grid minor;
%                    title({['Mean  =  '  num2str(mean(data.x))  ',  STD  =  '
num2str(std(data.x)) ];});
        end

    else
        warning(['In order to train network, please load data.' ...
        'Input data should be an excel file with only one column!']);
        disp('Operation Cancel.');
        data.isDataRead = false;
    end
end
function data = PreInput(data)
    if iscell(data)
        for i=1:size(data,1)
            for j=1:size(data,2)
                if strcmpi(data{i,j},'#NULL!')
                    tempVars(i,j) = NaN; %#ok
                else
                    tempVars(i,j) = str2num(data{i,j});   %#ok
                end
            end
        end
        data = tempVars;
    end

end
function  vars = DataStandardization(data)
    for i=1:size(data,2)
        x.mu(1,i)   = mean(data(:,i),'omitnan');
        x.sig(1,i)  = std (data(:,i),'omitnan');
        vars(:,i) = (data(:,i) - x.mu(1,i))./ x.sig(1,i);
    end
end
function vars = DataNormalization(data)
    for i=1:size(data,2)
        vars(:,i)   =   (data(:,i)   -min(data(:,i)))./   (max(data(:,i))-
min(data(:,i)));
    end
end
% --------------- Train Network ---------------
% ---------------------------------------------
function [opt,data] = TrainData(opt,data)
% Bi-LSTM parameters
if opt.NumOfHiddenLayers ==1
    opt.numHiddenUnits1 = opt.NumOfUnitsInFirstlayer;
elseif opt.NumOfHiddenLayers ==2
    opt.numHiddenUnits1 = opt.NumOfUnitsInFirstlayer;
    opt.numHiddenUnits2 = opt.NumOfUnitsInSecondlayer;
elseif opt.NumOfHiddenLayers ==3
    opt.numHiddenUnits1 = opt.NumOfUnitsInFirstlayer;
    opt.numHiddenUnits2 = opt.NumOfUnitsInSecondlayer;
    opt.numHiddenUnits3 = opt.NumOfUnitsInThirdlayer;
elseif opt.NumOfHiddenLayers ==4
    opt.numHiddenUnits1 = opt.NumOfUnitsInFirstlayer;
    opt.numHiddenUnits2 = opt.NumOfUnitsInSecondlayer;
    opt.numHiddenUnits3 = opt.NumOfUnitsInThirdlayer;
    opt.numHiddenUnits4 = opt.NumOfUnitsInFourthlayer;
end
% MLP parameters
if opt.NumOfFeedForwardLeyars ==1
    opt.ShallowhiddenLayerSize      =      [opt.NumOfNeuronsInFirstlayer];
% number of Hidden layers in MLP network.
elseif opt.NumOfFeedForwardLeyars ==2
    opt.ShallowhiddenLayerSize       =       [opt.NumOfNeuronsInFirstlayer
opt.NumOfNeuronsInSecondlayer];          % number of Hidden layers in MLP
network.
elseif opt.NumOfFeedForwardLeyars ==3
    opt.ShallowhiddenLayerSize       =       [opt.NumOfNeuronsInFirstlayer
opt.NumOfNeuronsInSecondlayer  opt.NumOfNeuronsInThirdlayer];%  number  of
Hidden layers in MLP network.
end



% prepare delays for time serie network
data = CreateTimeSeriesData(opt,data);

% divide data into test and train data
data = dataPartitioning(opt,data);

if strcmpi(opt.learningMethod,'LSTM')
    % LSTM data form
    data = LSTMInput(data);
    %  Define LSTM  architect
    opt = LSTMArchitect(opt,data);
elseif strcmpi(opt.learningMethod,'MLP')
    % Prepare input data for MLP network.
    FeedForwardInput();
    %  Define MLP architect
    opt   = FeedForwardArchitect(opt);
end

% Train LSTM, MLP
data = TrainNet(opt,data);

end
% make some delays on input filed
function data = CreateTimeSeriesData(opt,data)
    Delays = opt.Delays;

    x = data.x';
    T = size(x,2);

    MaxDelay = max(Delays);

    Range = MaxDelay+1:T;

    X= [];
    for d = Delays
        X=[X; x(:,Range-d)];
    end

    Y = x(:,Range);
    data.X  = X;
    data.Y  = Y;
```

```
end
% partitioning input data
function data = dataPartitioning(opt,data)
data.XTr   = [];
data.YTr   = [];
data.XTs   = [];
data.YTs   = [];

numTrSample = round(opt.trPercentage*size(data.X,2));

data.XTr   = data.X(:,1:numTrSample);
data.YTr   = data.Y(:,1:numTrSample);

data.XTs   = data.X(:,numTrSample+1:end);
data.YTs   = data.Y(:,numTrSample+1:end);

disp(['Time Series data divided to ' num2str(opt.trPercentage*100) '% Train
data and ' num2str((1-opt.trPercentage)*100) '% Test data']);
end
% Prepare input data for MLP network.
function FeedForwardInput()
    disp('Time Series data prepared as suitable MLP Input data.');
end
% Prepare input data for LSTM network.
function data = LSTMInput(data)

for i=1:size(data.XTr,2)
    XTr{i,1} = data.XTr(:,i);
    YTr(i,1) = data.YTr(:,i);
end

for i=1:size(data.XTs,2)
    XTs{i,1} =  data.XTs(:,i);
    YTs(i,1) =  data.YTs(:,i);
end
data.XTr   = XTr;
data.YTr   = YTr;
data.XTs   = XTs;
data.YTs   = YTs;

disp('Time Series data prepared as suitable LSTM Input data.');
end

% ---- network structure ----
% bi-LSTM Deeplearning Architect
function opt = LSTMArchitect(opt,data)

miniBatchSize    = opt.miniBatchSize;
maxEpochs        = opt.maxEpochs;
trainingProgress = opt.trainingProgress;
executionEnvironment = opt.executionEnvironment;

inputSize = size(data.X,1);
outputMode = 'last';
numResponses    = 1;
dropoutVal = opt.DropoutValue;
if opt.isUseDropoutLayer % if dropout layer is true
    if opt.NumOfHiddenLayers ==1
        if opt.isUseBiLSTMLayer == 1
            opt.layers = [ ...
```

```
                sequenceInputLayer(inputSize)
                bilstmLayer(opt.numHiddenUnits1,'OutputMode',outputMode)
                dropoutLayer(dropoutVal)
                fullyConnectedLayer(numResponses)
                regressionLayer];
        else
            opt.layers = [ ...
                sequenceInputLayer(inputSize)
                lstmLayer(opt.numHiddenUnits1,'OutputMode',outputMode)
                dropoutLayer(dropoutVal)
                fullyConnectedLayer(numResponses)
                regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==2
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits2,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            lstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            lstmLayer(opt.numHiddenUnits2,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==3
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits3,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits3,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==4
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits3,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits4,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits3,'OutputMode','sequence')
            dropoutLayer(dropoutVal)
            bilstmLayer(opt.numHiddenUnits4,'OutputMode',outputMode)
            dropoutLayer(dropoutVal)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    end
else % if dropout layer is false
    if opt.NumOfHiddenLayers ==1
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
                sequenceInputLayer(inputSize)
                bilstmLayer(opt.numHiddenUnits1,'OutputMode',outputMode)
                fullyConnectedLayer(numResponses)
                regressionLayer];
        else
            opt.layers = [ ...
                sequenceInputLayer(inputSize)
                lstmLayer(opt.numHiddenUnits1,'OutputMode',outputMode)
                fullyConnectedLayer(numResponses)
                regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==2
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits2,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            lstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            lstmLayer(opt.numHiddenUnits2,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==3
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits3,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits3,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    elseif opt.NumOfHiddenLayers ==4
        if opt.isUseBiLSTMLayer
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits3,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits4,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        else
            opt.layers = [ ...
            sequenceInputLayer(inputSize)
            bilstmLayer(opt.numHiddenUnits1,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits2,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits3,'OutputMode','sequence')
            bilstmLayer(opt.numHiddenUnits4,'OutputMode',outputMode)
            fullyConnectedLayer(numResponses)
            regressionLayer];
        end
    end
end
% Training Network Options
% 'sgdm'
% 'rmsprop'
% 'adam'

opt.opts = trainingOptions(opt.LR, ...
    'MaxEpochs',maxEpochs, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.005, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod',125, ...
    'LearnRateDropFactor',0.2, ...
    'Verbose',1, ...
    'MiniBatchSize',miniBatchSize,...
    'ExecutionEnvironment',executionEnvironment,...
    'Plots',trainingProgress);
    disp('LSTM architect successfully created.');
end
% MLP Shallowlearning Architect
function opt = FeedForwardArchitect(opt)
opt.Net = feedforwardnet(opt.ShallowhiddenLayerSize,opt.trainFcn);
opt.Net.divideParam.trainRatio = 80/100;
opt.Net.divideParam.valRatio  = 10/100;
opt.Net.divideParam.testRatio = 10/100;
opt.Net.trainParam.epochs          = opt.maxItrations;
opt.Net.trainParam.showWindow      = opt.showWindow;
opt.Net.trainParam.showCommandLine = opt.showCommandLine;
disp('MLP architect successfully created.');

end
% Train  Network
function data = TrainNet(opt,data)

if strcmpi(opt.learningMethod,'LSTM')
    try
        data.BiLSTM.Net                                                  =
trainNetwork(data.XTr,data.YTr,opt.layers,opt.opts);
        disp('LSTM Netwwork successfully trained.');
        data.IsNetTrainSuccess =true;
    catch me
        disp('Error on Training LSTM Network');
        data.IsNetTrainSuccess = false;
        return;
    end
elseif strcmpi(opt.learningMethod,'MLP')
    try
        [data.FF.Net,~] = train(opt.Net,data.XTr,data.YTr);
        disp('Feed Forward Netwwork successfully trained.');
        data.IsNetTrainSuccess = true;
    catch me
        disp('Error on Training FF Network');
        data.IsNetTrainSuccess =false;
        return;
    end
end

end


% --------------- Evaluate Data ---------------
% ---------------------------------------------
function [opt,data] = EvaluationData(opt,data)
if strcmpi(opt.learningMethod,'LSTM')

    data.BiLSTM.TrainOutputs                                             =
deNorm(data.seriesdata,predict(data.BiLSTM.Net,data.XTr,'MiniBatchSize',o
pt.miniBatchSize),opt.dataPreprocessMode);
    data.BiLSTM.TrainTargets                                             =
deNorm(data.seriesdata,data.YTr,opt.dataPreprocessMode);
    data.BiLSTM.TestOutputs                                              =
deNorm(data.seriesdata,predict(data.BiLSTM.Net,data.XTs,'MiniBatchSize',o
pt.miniBatchSize),opt.dataPreprocessMode);
    data.BiLSTM.TestTargets                                              =
deNorm(data.seriesdata,data.YTs,opt.dataPreprocessMode);
    data.BiLSTM.AllDataTargets         =         [data.BiLSTM.TrainTargets
data.BiLSTM.TestTargets];
    data.BiLSTM.AllDataOutputs         =         [data.BiLSTM.TrainOutputs
data.BiLSTM.TestOutputs];

%     data = PlotResults(data,'Tr',...
%         data.BiLSTM.TrainOutputs, ...
%         data.BiLSTM.TrainTargets);
%                                                          data          =
plotReg(data,'Tr',data.BiLSTM.TrainTargets,data.BiLSTM.TrainOutputs);
```

```
%
%     data = PlotResults(data,'Ts',....
%         data.BiLSTM.TestOutputs, ...
%         data.BiLSTM.TestTargets);
%                                                         data        =
plotReg(data,'Ts',data.BiLSTM.TestTargets,data.BiLSTM.TestOutputs);
%
%     data = PlotResults(data,'All',...
%         data.BiLSTM.AllDataOutputs, ...
%         data.BiLSTM.AllDataTargets);
%                                                         data        =
plotReg(data,'All',data.BiLSTM.AllDataTargets,data.BiLSTM.AllDataOutputs)
;
%
    disp('Bi-LSTM network performance evaluated.');

elseif strcmpi(opt.learningMethod,'MLP')

        data.FF.TrainOutputs                                              =
deNorm(data.seriesdata,data.FF.Net(data.XTr)',opt.dataPreprocessMode);
        data.FF.TrainTargets                                              =
deNorm(data.seriesdata,(data.YTr)',opt.dataPreprocessMode);
        data.FF.TestOutputs                                               =
deNorm(data.seriesdata,data.FF.Net(data.XTs)',opt.dataPreprocessMode);
        data.FF.TestTargets                                               =
deNorm(data.seriesdata,(data.YTs)',opt.dataPreprocessMode);

    data.FF.AllDataTargets = [data.FF.TrainTargets data.FF.TestTargets];
    data.FF.AllDataOutputs = [data.FF.TrainOutputs data.FF.TestOutputs];

    DispVal = 1;
    for i= DispVal
%         data = PlotResults(data,'Tr',...
%             data.FF.TrainOutputs(i,:), ...
%             data.FF.TrainTargets(i,:));
%                                                             data      =
plotReg(data,'Tr',data.FF.TrainTargets(i,:),data.FF.TrainOutputs(i,:));
%         data = PlotResults(data,'Ts',....
%             data.FF.TestOutputs(i,:), ...
%             data.FF.TestTargets(i,:));
%                                                             data      =
plotReg(data,'Ts',data.FF.TestTargets(i,:),data.FF.TestOutputs(i,:));
%         data = PlotResults(data,'All',...
%             data.FF.AllDataOutputs(i,:), ...
%             data.FF.AllDataTargets(i,:));
%                                                             data      =
plotReg(data,'All',data.FF.AllDataTargets(i,:),data.FF.AllDataOutputs(i,:
));
        disp('MLP network performance evaluated.');
    end

end
end
function vars = deNorm(data,stdData,deNormMode)
if iscell(stdData(1,1))
        for i=1:size(stdData,1)
            tmp(i,:) = stdData{i,1}';
        end
        stdData = tmp;
end
```

```
if strcmpi(deNormMode,'Data Normalization')
    for i=1:size(data,2)
        vars(:,i)  =  (stdData(:,i).*(max(data(:,i))-min(data(:,i))))  +
min(data(:,i));
    end
    vars = vars';

elseif strcmpi(deNormMode,'Data Standardization')
    for i=1:size(data,2)
        x.mu(1,i)   = mean(data(:,i),'omitnan');
        x.sig(1,i)  = std (data(:,i),'omitnan');
        vars(:,i) = ((stdData(:,i).* x.sig(1,i))+ x.mu(1,i));
    end
    vars = vars';

else
    vars = stdData';
    return;
end
end
% plot the output of networks and real output on test and train data
function data = PlotResults(data,firstTitle,Outputs,Targets)
Errors = Targets - Outputs;
    MSE   = mean(Errors.^2);
    RMSE  = sqrt(MSE);
    NRMSE = RMSE/mean(Targets);
    ErrorMean = mean(Errors);
    ErrorStd  = std(Errors);
    rankCorre = RankCorre(Targets,Outputs);

    if strcmpi(firstTitle,'tr')
         Disp1Name = 'OutputGraphEvaluation_TrainData';
         Disp2Name = 'ErrorEvaluation_TrainData';
         Disp3Name = 'ErrorHistogram_TrainData';
    elseif strcmpi(firstTitle,'ts')
        Disp1Name = 'OutputGraphEvaluation_TestData';
        Disp2Name = 'ErrorEvaluation_TestData';
        Disp3Name = 'ErrorHistogram_TestData';
    elseif strcmpi(firstTitle,'all')
        Disp1Name = 'OutputGraphEvaluation_ALLData';
        Disp2Name = 'ErrorEvaluation_ALLData';
        Disp3Name = 'ErrorHistogram_AllData';
    end

    figure('Name',Disp1Name,'NumberTitle','off');
    plot(1:length(Targets),Targets,...
        1:length(Outputs),Outputs);grid minor
    legend('Targets','Outputs','Location','best') ;
    title(['Rank Correlation = ' num2str(rankCorre)]);

    figure('Name',Disp2Name,'NumberTitle','off');
    plot(Errors);grid minor
    title({['MSE = ' num2str(MSE) ', RMSE = ' num2str(RMSE)...
        ' NRMSE = ' num2str(NRMSE)] ;});
    xlabel(['Error Per Sample']);

    figure('Name',Disp3Name,'NumberTitle','off');
    histogram(Errors);grid minor
    title(['Error  Mean  =  '  num2str(ErrorMean)  ',  Error  StD  =  '
num2str(ErrorStd)]);
    xlabel(['Error Histogram']);

    if strcmpi(firstTitle,'tr')
        data.Err.MSETr = MSE;
        data.Err.STDTr = ErrorStd;
        data.Err.NRMSETr     = NRMSE;
        data.Err.rankCorreTr = rankCorre;
    elseif strcmpi(firstTitle,'ts')
        data.Err.MSETs = MSE;
        data.Err.STDTs = ErrorStd;
        data.Err.NRMSETs     = NRMSE;
        data.Err.rankCorreTs = rankCorre;
    elseif strcmpi(firstTitle,'all')
        data.Err.MSEAll = MSE;
        data.Err.STDAll = ErrorStd;
        data.Err.NRMSEAll     = NRMSE;
        data.Err.rankCorreAll = rankCorre;
    end
end
% find rank correlation between network output and real data
function [r]=RankCorre(x,y)
x=x';
y=y';
% Find the data length
N = length(x);
% Get the ranks of x
R = crank(x)';
for i=1:size(y,2)
    % Get the ranks of y
    S = crank(y(:,i))';
    % Calculate the correlation coefficient
    r(i) = 1-6*sum((R-S).^2)/N/(N^2-1); %#ok
end
end
function r=crank(x)
u = unique(x);
[~,z1] = sort(x);
[~,z2] = sort(z1);
r = (1:length(x))';
r=r(z2);
for i=1:length(u)
    s=find(u(i)==x);
    r(s,1) = mean(r(s));
end
end
% plot the regression line of output and real value
function data = plotReg(data,Title,Targets,Outputs)

 if strcmpi(Title,'tr')
     DispName = 'RegressionGraphEvaluation_TrainData';
elseif strcmpi(Title,'ts')
    DispName = 'RegressionGraphEvaluation_TestData';
elseif strcmpi(Title,'all')
    DispName = 'RegressionGraphEvaluation_ALLData';
end
figure('Name',DispName,'NumberTitle','off');
x = Targets';
y = Outputs';
```

```
format long
b1 = x\y;
yCalc1 = b1*x;
scatter(x,y,'MarkerEdgeColor',[0 0.4470 0.7410],'LineWidth',.7);
hold('on');
plot(x,yCalc1,'Color',[0.8500 0.3250 0.0980]);
xlabel('Prediction');
ylabel('Target');
grid minor
% xgrid = 'on';
% disp.YGrid = 'on';
X = [ones(length(x),1) x];
b = X\y;
yCalc2 = X*b;
plot(x,yCalc2,'-.','MarkerSize',4,"LineWidth",.1,'Color',[0.9290   0.6940
0.1250])
legend('Data','Fit','Y=T','Location','best');
%
Rsq2 = 1 -  sum((y - yCalc1).^2)/sum((y - mean(y)).^2);

if strcmpi(Title,'tr')
    data.Err.RSqur_Tr = Rsq2;
    title(['Train Data, R^2 = ' num2str(Rsq2)]);
elseif strcmpi(Title,'ts')
    data.Err.RSqur_Ts = Rsq2;
    title(['Test Data, R^2 = ' num2str(Rsq2)]);
elseif strcmpi(Title,'all')
    data.Err.RSqur_All = Rsq2;
    title(['All Data, R^2 = ' num2str(Rsq2)]);
end

end
end
```