Connected Coverage Assurance for Sensor Scheduling in Wireless Sensor Networks

Attapol ADULYASAS

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

Institute for Communication Systems
Department of Electronic Engineering
Faculty of Engineering and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, UK

December 2015

© Attapol ADULYASAS, 2015
In the Name of Allah,

the Most Gracious and the Most merciful
Abstract

A long operational lifetime is one of ultimate goals of wireless sensor networks (WSNs) due to the limited energy resources of sensors. As sensors are often randomly deployed in vast and inaccessible areas, it is impractical to recharge or replace their energy resources such as batteries. Thus, energy efficiency is likely to be a highly important issue for the WSNs.

A key approach to enhancing energy efficiency is sensor scheduling. Sensor scheduling means that in each operational round certain sensors are selected to be active, whilst others are pushed into sleep mode. However, the required quality of sensing coverage and network connectivity must be guaranteed. The former is that the entire monitored area must be fully covered at a given level called a desired coverage degree ($k$). Meanwhile, the latter is that every active sensor must be connected with others. Both properties together are known as the connected coverage assurance.

This thesis proposes a series of sensor scheduling methods, namely 6-Triangle ($6$-Tri), 4-Square ($4$-Sqr), 3-Symmetrical area ($3$-Sym) and Optimum-Symmetrical area ($O$-Sym). The $6$-Tri method uses a hexagon tessellation as a virtual partition in order to group sensors into hexagonal cells. This method activates 6 sensors from each cell. Otherwise, the $4$-Sqr method uses a virtual square partition instead in order to divide the sensors into square cells. A cell consists of 4 sub-squares, within each of which a sensor is activated.

Similar to the $6$-Tri method, the $3$-Sym method has a hexagon tessellation as its virtual partition. As only three sensors whose position is symmetrical with each other are selected in each cell, the $3$-Sym method can significantly reduce the number of active sensors, compared to both the $6$-Tri and $4$-Sqr methods. The $O$-Sym method enhances the $3$-Sym method by optimising coverage efficiency. It firstly investigates coverage redundancy, which is produced by the $3$-Sym method, and then tries to minimise the redundancy to the desired coverage degree. This method achieves both energy efficiency and coverage efficiency, which are the main objectives of this thesis.

Keywords: sensor scheduling, connected coverage assurance, virtual partition, and WSNs

Email: a.adulyasas@surrey.ac.uk
Acknowledgements

Firstly, I would like to acknowledge the support of the Strategic Scholarships Frontier Research Networks (Specific for Southern region) for the Ph.D. programme by the Office of the Higher Education Commission, Thailand.

I also would like to express my deep gratitude to my principal supervisor, Prof. Zhili Sun, for his valuable guidance, support, and kindness. In addition, my appreciation goes to my co-supervisor, Dr. Ning Wang, for his constructive and insightful suggestions. I would like to thank my colleagues and the staff at the Institute for Communication Systems (ICS) for their comments, advice, and friendly support. I wish to thank all of the International Student Support team for their suggestions and the help. I would like to extend my thanks to the University of Surrey, especially the Department of Electronic Engineering for providing resources and an excellent environment throughout my Ph.D. study.

I would like also to express my thanks to Asst. Prof. Dr. Sombat Yothatip, the President and Asst. Prof. Kraisorn Sritairat, the former President, Asst. Prof. Abdulnaser Hajisamoh, the Dean of the Faculty of Science Technology and Agriculture, and all of my colleagues at Yala Rajabhat University, Thailand, for their encouragement and support.

Major recognition goes to my mother for her love, encouragement, and the sacrifices that she has made for me throughout her life. Special gratitude is extended to my family, who support me and make my life stable and warm.

Finally, all praise is due to my lord, Allah, who is most compassionate and most merciful. This work indeed could not have been achieved at all without his allowance. My life and my soul are for the sake of Allah.
# Table of Contents

Abstract ........................................................................................................................................... iii

Acknowledgements .................................................................................................................................. iv

Table of Contents ..................................................................................................................................... v

List of Publications ................................................................................................................................... ix

List of Figures .......................................................................................................................................... x

List of Tables ........................................................................................................................................... xiv

List of Symbols ......................................................................................................................................... xv

Glossary of Terms ................................................................................................................................... xviii

Chapter 1 Introduction ............................................................................................................................... 1
  1.1 Background ........................................................................................................................................ 1
  1.2 Motivation ........................................................................................................................................... 3
  1.3 Research Objectives ............................................................................................................................ 4
  1.4 Main Contributions .............................................................................................................................. 5
  1.5 Thesis Structure ................................................................................................................................... 7

Chapter 2 Background and State of the Art ............................................................................................. 9
  2.1 The Connected Coverage Problems ................................................................................................ 9
    2.1.1 Discrete point connected coverage problem (DPCC) ................................................................. 9
    2.1.2 Partial area connected coverage problem (PACC) ................................................................. 10
    2.1.3 Entire area connected coverage problem (EACC) ................................................................. 13
  2.2 Relevant Properties ............................................................................................................................. 14
    2.2.1 Sensor properties ......................................................................................................................... 14
    2.2.2 Sensing coverage properties ....................................................................................................... 15
    2.2.3 Connectivity properties .............................................................................................................. 19
    2.2.4 Algorithm properties .................................................................................................................. 21
  2.3 Connected Coverage Verification and Control ................................................................................ 22
2.3.1 Coverage control strategies ................................................................. 22
2.3.2 Connectivity control strategies ............................................................ 25

2.4 Sensor Scheduling .................................................................................. 26

2.5 Evaluation metrics ................................................................................... 28
2.5.1 Complexities of algorithms ................................................................. 28
2.5.2 Constraints of algorithms ................................................................. 28
2.5.3 Quality of the connected coverage .................................................... 29
2.5.4 Energy efficiency ............................................................................... 30
2.5.5 Fault tolerance .................................................................................. 30

2.6 The State of the Art .................................................................................. 31
2.6.1 Geographical based strategy ............................................................... 31
2.6.2 Heuristic based strategy .................................................................... 36
2.6.3 Stochastic based strategy ................................................................... 39
2.6.4 Partition based strategy ................................................................. 43

2.7 Summary .................................................................................................. 46

Chapter 3 A Virtual Hexagon Partition for Achieving Connected Coverage

Assurance under Sensor Scheduling ............................................................. 49

3.1 Introduction .............................................................................................. 49
3.1.1 Objectives ......................................................................................... 50
3.1.2 Contributions .................................................................................. 50
3.1.3 Chapter organisation ......................................................................... 50

3.2 Network Model ......................................................................................... 51

3.3 A Virtual Hexagon Partition and Position Determination ......................... 51
3.3.1 Fundamental hexagonal cell ............................................................... 52
3.3.2 Position determination ...................................................................... 53

3.4 6-Triangle (6-Tri) Sensor Scheduling Method ............................................. 55
3.4.1 Centre method .................................................................................. 56
3.4.2 6 Triangles (6-Tri) method ............................................................... 57
3.4.3 Sensor scheduling ............................................................................. 58
3.4.4 Sensor selection algorithm ............................................................... 58

3.5 Analyses and Simulations ......................................................................... 60
3.5.1 Analyses .......................................................................................... 60
3.5.2 Simulations ....................................................................................... 62
Chapter 4 A Virtual Square Partition for Achieving Connected Coverage

4.1 Introduction.................................................................68
4.1.1 Objectives.............................................................69
4.1.2 Contributions.........................................................69
4.1.3 Chapter organisation ...............................................70
4.2 A Virtual Square Partition and Position Determination ................70
4.2.1 Fundamental square cell and virtual partition ...................70
4.2.2 Position determination ..............................................72
4.3 4-Square (4-Sqr) Sensor Scheduling Method .....................73
4.3.1 4-Sqr method .......................................................74
4.3.2 Processes of the 4-Sqr method ...................................74
4.3.3 Sensor selection algorithm .......................................75
4.4 Relaxing Relationship between Communication and Sensing Radii ....76
4.5 Analyses and Simulations ...........................................77
4.5.1 Chance of being selected........................................77
4.5.2 Success of sensor selection .......................................78
4.5.3 Number of active sensors ......................................79
4.5.4 Quality of the connected coverage .............................81
4.5.5 Network’s lifetime ...................................................86
4.6 Summary ......................................................................87

Chapter 5 Optimisation of Sensor Scheduling Methods for Connected Coverage

5.1 Introduction.................................................................89
5.1.1 Objectives .............................................................89
5.1.2 Contributions .........................................................90
5.1.3 Chapter organisation ...............................................91
5.2 Optimum Number of Active Sensors in a Hexagonal Cell ..........91
5.2.1 One sensor per covering a hexagonal cell .......................91
5.2.2 Two sensors per covering a hexagonal cell ......................92
5.2.3 Three sensors per covering a hexagonal cell ....................95
List of Publications

Journal Paper


Conference Papers


List of Figures

Figure 1-1: A wireless sensor network and architecture of a sensor node .........................................................2

Figure 2-1: Discrete point coverage problem ........................................................................................................10

Figure 2-2: Partial (barrier) area coverage problem ...........................................................................................11

Figure 2-3: An example of the entire area coverage (EAC) ................................................................................13

Figure 2-4: Illustration of (a) binary disc coverage model (b) probabilistic coverage model (c) binary sector coverage model .................................................................................................................16

Figure 2-5: Detection probability of the probabilistic coverage model with different parameters $\lambda$ and $\beta$ .................................................................................................................................17

Figure 2-6: Sensing cover degrees of (a) disc coverage model (b) sector coverage model ........................................18

Figure 2-7: Relationship between the communication and sensing radii ............................................................20

Figure 2-8: Bottleneck of the critical sensor .........................................................................................................21

Figure 2-9: Grid points marked and counted by a sensor to obtain the covered area ........................................23

Figure 2-10: Illustration of both covered intersection and uncovered intersection points .......................................23

Figure 2-11: Illustration of perimeter coverage ..................................................................................................24

Figure 2-12: Coverage implying connectivity by relationship $R_c \geq 2R_s$ .............................................................25

Figure 2-13: Sensor scheduling by certain duration of the operational round .........................................................27

Figure 2-14: Investigation of intersection point $O$ and selection a sensor being closet to point $P$ ................................32

Figure 2-15: The target points $P_1$, $P_2$ and $P_3$ being close to each other .........................................................34

Figure 2-16: Lens area used in the Lens method ..................................................................................................43
Figure 2-18: Sponsors and covering area on a target area.................................44
Figure 2-17: The triangle tessellation layered around a cluster head ....................44
Figure 3-1: A hexagonal cell.............................................................................52
Figure 3-2: The virtual hexagon partition (quadrant 1)....................................53
Figure 3-3: Four possible cases of two candidates of hexagonal cells in form column-row ........................................................................................................54
Figure 3-4: Chance of sensor selection...............................................................63
Figure 3-5: Success of sensor selection ...............................................................64
Figure 3-6: Number of active sensors vs. network density...............................65
Figure 3-7: Number of active sensors vs. \( R_s \) ..................................................65
Figure 3-8: Number of active sensors vs. \( \mathcal{A}/\mathcal{A}_{\text{tar}} \) ................................66
Figure 3-9: Quality of coverage vs. network density.........................................67
Figure 4-1: The square cell \( \mathcal{S}(O_{uv}, P_z, R_s) \) ..............................................70
Figure 4-2: The structure of the virtual square partition (a) Partition in quadrant 1 (b) Four cell candidates of any sensor, \( (O_a, O_b, O_c, O_d) \) ..........................71
Figure 4-3: Chance of sensors being selected....................................................78
Figure 4-4: Success of sensors selection.............................................................79
Figure 4-5: Number of active sensors vs. density.............................................80
Figure 4-6: Number of active sensors vs. \( R_c/R_s \) ..............................................80
Figure 4-7: Number of active sensors vs. sensing radius...............................81
Figure 4-8: Quality of connectivity vs. \( R_s \) ......................................................83
Figure 4-9: Quality of connectivity vs. \( R_c/R_s \) ..............................................83
Figure 4-11: Quality of coverage vs. $R_c/R_s$ ................................................................. 84
Figure 4-10: Quality of coverage vs. density ................................................................. 84
Figure 4-12: Quality of coverage vs position error ....................................................... 85
Figure 4-13: Network lifetime ....................................................................................... 86
Figure 4-14: Connected coverage vs network lifetime ............................................... 87
Figure 4-15: Average of energy consumption ............................................................. 87
Figure 5-1: Examples of connected coverage by two nodes for a cell ..................... 93
Figure 5-2: The perfect symmetrical positions of three sensors .................................. 96
Figure 5-3: The almost symmetry (symmetrical angles and different distances) of three sensors ........................................................................................................ 97
Figure 5-4: The almost symmetry (symmetrical distances and different angles) of covering a hexagonal cell by three sensors ......................................................... 98
Figure 5-5: Three symmetrical areas ........................................................................... 101
Figure 5-6: 3-Sym method ......................................................................................... 105
Figure 5-7: Territory of an active sensor composed of three pieces of equilateral triangles. The territory’s possessor is in the middle piece ........................................ 110
Figure 5-8: Critical points made .................................................................................. 112
Figure 5-9: Examples of incomplete coverage ............................................................ 113
Figure 5-10: Chance of sensors being selected ............................................................ 121
Figure 5-11: Success of sensor selection ...................................................................... 122
Figure 5-12: Number of active sensors vs. density ..................................................... 123
Figure 5-13: Number of active sensors vs. $R_s$ .......................................................... 124
Figure 5-14: Number of active sensors vs. $R_C/R_S$ ................................................................. 124

Figure 5-15: Connected coverage vs. density ................................................................. 125

Figure 5-16: Connectivity vs. $R_S$ ........................................................................... 126

Figure 5-17: Connected coverage vs. position error ...................................................... 127

Figure 5-18: Proportion of coverage efficiency ............................................................... 128

Figure 5-20: Energy consumption and load balance ....................................................... 129

Figure 5-19: Network lifetime ....................................................................................... 129

Figure 5-21: Connected coverage vs. network lifetime .............................................. 130
List of Tables

Table 2-1: Energy Consumption Values of Sensor in Different Duties ................................................. 27

Table 2-2: Achievements and Drawbacks of the Geographical based Approaches ......................... 36

Table 2-3: Achievements and Drawbacks of the Heuristic based Approaches ................................. 39

Table 2-4: Achievements and Drawbacks of the Stochastic based Approaches ............................... 42

Table 2-5: Achievements and Drawbacks of the Partition Based Approaches .................................. 46

Table 2-6: Comparison of the Approaches in the Connected Coverage Assurance for Sensor Scheduling .............................................................................................................................................. 47

Table 2-7: Properties of the Approaches in the Connected Coverage Assurance for Sensor Scheduling .................................................................................................................................................... 48

Table 3-1: Tools of Simulation and Network Scenarios ........................................................................ 63

Table 4-1: Energy Parameters used in Simulation ................................................................................. 85

Table A–1: Hexagon Partition .............................................................................................................. 139

Table A–2: Sensor Selection of the 4-Sqr Method .............................................................................. 141

Table B–1: Square Partition ................................................................................................................ 143

Table B–2: Sensor Section of the 4-Sqr Method .................................................................................. 145

Table C– 1: Sensor Selection of the 3-Sym Method ............................................................................. 147

Table D– 1: Sensor Selection of the O-Sym Method ............................................................................ 151
List of Symbols

\( \mathcal{A} \)  
Area of a monitored field and its dimension is \( \mathcal{A} = W \times L \)

\( C(\cdot) \)  
Coverage probability

\( Ch \)  
Chance of sensors being selected

\( \mathcal{E}(\cdot) \)  
Equivalent sensing disc

\( c(\cdot) \)  
Function of the binary disc coverage model

\( \mathcal{D}(s_i, R_s) \) or \( \mathcal{D}_i \)  
Sensing disc of sensor \( s_i \)

\( d(p, q) \)  
Euclidean distance between point \( p \) and point \( q \)

\( E[\cdot] \)  
Expected coverage ratio

\( E_i \)  
Initial energy of sensor \( s_i \)

\( e_i \)  
Residual energy of sensor \( s_i \)

\( e_{ij} \)  
Energy consumed by sensor \( s_i \) to transmit a bit of data to sensor \( s_j \)

\( \mathcal{H}(\cdot) \)  
Hexagonal cell

\( k \)  
Desired coverage degree

\( L \)  
Length of a monitored field

\( \mathcal{L} \)  
Lens area

\( N \)  
Total number of sensors in a network

\( N_a \)  
Number of active sensors

\( N_{ant} \)  
Number of ants

\( N_{as} \)  
Number of assignments

\( N_D \)  
Number of hops of network dimension

\( N_{DPOI} \)  
Number of discrete points of interest

\( N_i \)  
Number of neighbours of sensor \( s_i \)

\( N_{it} \)  
Number of iterations

\( N_r \)  
Number of sensors performing the relaying task

\( N_s \)  
Number of sensor performing the sensing task

\( n_j^i \)  
Neighbour \( j \) of sensor \( s_i \)

\( O_{uv} \)  
Centre of a cell at \((u, v)\) alignments

\( P \)  
Set of discrete points of interest (DPOIs) where \( P = \{ p_m | m = 1, 2, \ldots, N_{DPOI} \} \)
\( P_r \)  
Received power

\( P_t \)  
Transmitted power

\( P_z \)  
Piece of target area in a cell where \( \{ z = 1, 2, 3, ..., Z \} \)

\( p_m \)  
Point of interest

\( p_c(i) \)  
Critical point made by sensor \( s_i \)

\( p_{cg}^i(s_i) \)  
Critical point on an edge of a territory

\( p_i \)  
Probability of sensor \( s_i \) being selected

\( q_{cg}^i(s_i, s_j) \)  
Critical point inside the territory

\( R_c \)  
Communication radius

\( R_s \)  
Sensing radius

\( S \)  
Set of sensors where \( (S = \{ s_i | i = 1, 2, ..., N \}) \)

\( S(\cdot) \)  
Square cell

\( s_D \)  
Dependent node

\( s_i \)  
Independent node

\( s_i \)  
Sensor \( s_i \)

\( T \)  
Network lifetime

\( T(s_i) \)  
Territory of sensor \( s_i \)

\( t_p \)  
Period of probing

\( t_s \)  
Time slot

\( W \)  
Width of a monitored field

\( \mathcal{W} \)  
Proportion between the number of the uncovered target points and the path weight

\( w_{ij} \)  
Link weight between sensors \( i \) and \( j \)

\( \Gamma \)  
Set of assignments where \( \{ \Gamma | i = 1, 2, 3, ..., N_{ass} \} \)

\( \gamma \)  
Pheromone trail

\( \delta \)  
Heuristic desirability

\( \eta \)  
Success of sensor selection

\( \theta_c \)  
Critical angle

\( \lambda \)  
Coverage efficiency

\( \mu \)  
Actual coverage degree

\( \rho \)  
Network density

\( \sigma \)  
Path loss exponent
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_i$</td>
<td>Operational time of sensor $s_i$</td>
</tr>
<tr>
<td>$\lambda^\phi$</td>
<td>Proportion of coverage efficiency</td>
</tr>
<tr>
<td>$\Pi$</td>
<td>Proportion of covered area</td>
</tr>
<tr>
<td>$\phi$</td>
<td>Quality of connected coverage</td>
</tr>
<tr>
<td>$\Psi$</td>
<td>Quality of connectivity</td>
</tr>
<tr>
<td>$\Omega^k$</td>
<td>Quality of $k$-coverage</td>
</tr>
</tbody>
</table>
# Glossary of Terms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D</td>
<td>Two-Dimension</td>
</tr>
<tr>
<td>3-D</td>
<td>Three-Dimension</td>
</tr>
<tr>
<td>ACB-SA</td>
<td>Ant-Colony-Based Scheduling Algorithm</td>
</tr>
<tr>
<td>ACO</td>
<td>Ant Colony Optimisation</td>
</tr>
<tr>
<td>ACO-MNCC</td>
<td>ACO for Maximizing the Number of Connected Covers</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>BS</td>
<td>Base Station</td>
</tr>
<tr>
<td>COR</td>
<td>Coverage-Optimising Recursive</td>
</tr>
<tr>
<td>CSR</td>
<td>Communication-Sensing Relationship</td>
</tr>
<tr>
<td>CSS</td>
<td>Cyclic Segment Sequence</td>
</tr>
<tr>
<td>CWGC</td>
<td>Communication Weighted Greedy Cover</td>
</tr>
<tr>
<td>DBCCCR</td>
<td>Demand-Based Coverage and Connectivity-Preserving Routing Protocol</td>
</tr>
<tr>
<td>DCS</td>
<td>Deterministic Coverage Selection</td>
</tr>
<tr>
<td>DPCC</td>
<td>Discrete Point Connected Coverage problem</td>
</tr>
<tr>
<td>DPOI</td>
<td>Discrete Points of Interest</td>
</tr>
<tr>
<td>EACC</td>
<td>Entire Area Connected Coverage problem</td>
</tr>
<tr>
<td>EB</td>
<td>Energy Balance</td>
</tr>
<tr>
<td>EC</td>
<td>Energy Conservation</td>
</tr>
<tr>
<td>GCV</td>
<td>Generic Coverage Verification</td>
</tr>
<tr>
<td>GPS</td>
<td>Global Positioning System</td>
</tr>
<tr>
<td>I/O</td>
<td>Input and Output</td>
</tr>
<tr>
<td>LC</td>
<td>Location</td>
</tr>
<tr>
<td>LET</td>
<td>Location Error Tolerance</td>
</tr>
<tr>
<td>LP</td>
<td>Linear Program</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control</td>
</tr>
<tr>
<td>MC</td>
<td>Message Complexity</td>
</tr>
<tr>
<td>MCT</td>
<td>Maximum Cover Tree</td>
</tr>
<tr>
<td>OCU</td>
<td>Offline Cover Update</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PACC</td>
<td>Partial Area Connected Coverage problem</td>
</tr>
</tbody>
</table>
PLB  Probabilistic Load-Balancing
PRA  Probabilistic Redundancy Analysis
QCN  Quality of Connectivity
QCV  Quality of Coverage
ROC  Resilient Online Coverage
SC   Sensor Capability
SCPCONF Sensor Configuration Phase
SFA  Sensor Failure Avoidance
SINR Signal to Interference plus Noise Ratio
SYN  Synchronisation
TA   Two Anchors
TC   Time Complexity
TPACO Three Pheromones ACO
TSP  Traveling Salesman Problem
WSN(s) Wireless Sensor Network(s)
Chapter 1

Introduction

In this chapter, a brief background of wireless sensor networks is introduced. In addition, an overview of the thesis, including the research motivation, objectives and contributions are expressed.

1.1 Background

Wireless sensor networks (WSNs) have been researched for a decade and have been drawing increasing attention across the world in recent years [1]. A WSN comprises a large number of tiny, inexpensive, intelligent and autonomous sensor devices that are deployed in a monitored field [2]. The sensor devices can basically sense phenomena in the environment and can communicate with other associated sensors. In particular, they are intelligent enough to organise themselves as a network [3]. WSNs have therefore been adopted and have had a significant influence on numerous applications such as military surveillance [4] [5], forest fire surveillance [6] [7] [8], health monitoring and precision agriculture [9] [10] [11] [12].

A WSN is different from traditional wireless networks e.g., a wireless cellular network or a wireless LAN, as the sensors are capable of communicating directly with each other without any fixed infrastructure being facilitated. WSNs are distributed and self-organised networks in which the sensors can be deployed in either a pre-determined or random fashion particularly in inaccessible areas. The sensors, which are deployed across a monitored field, individually sense various kinds of physical properties and cooperatively deliver the sensed data through either direct or multi-hop communication towards a base station (BS) as shows in Figure 1-1. The BS typically has an unlimited power source and a high level of computational capability. It can therefore collect a huge amount of sensed data continuously from the sensors in the monitored field and it can quickly analyse the data. The information, which is an outcome of the analysis, will be used by users who generally have remote access to the BS through the internet.
Chapter 1. Introduction

The structure of a sensor node consists of four main parts, Sensing, Processing, Transceiver and Power units as shows in Figure 1-1, and two other optional parts, Location finding and Mobiliser units. The sensing unit is the tasking part. It undertakes periodic sampling and measures the physical analogue signals or any target that the applications prefer to monitor. Then the measured value is converted to digital data by an Analogue-to-Digital Convertor (ADC). Any instruction will be executed at the processing unit where the essential software is installed. The software consists of two parts: (i) the operating system (OS), which manages the node hardware; and (ii) the communication programme, which manages the connectivity among the sensors. The sensors can communicate with each other through the transceiver, which is composed of a transmitter, receiver and antenna. All electronic circuits of units consume energy provided by the power unit, which is typically a set of batteries. A location finding system e.g., a Global Positioning System (GPS) is required in some applications to identify the sensors’ locations. Similarly, mobility of sensors may be required in order to re-
locate sensors after deployment, so a mobiliser unit is an optional component according to the application requirements.

There are some principal characteristics that should be considered in order to understand the features and constraints of WSNs. Firstly, the sensors are typically deployed with high density in a vast area such as battle fields [4] [5], mountains and forests [6] [7], cultivated fields [9] [10]. These large quantities of sensors in a WSN can cause a high rate of data collision and a large amount of overheads. Secondly, the topology in a WSN changes frequently due to factors such as weather, quality of the wireless connection and sensor failure. Finally, the sensors typically have limited energy resources and memories, and they are also limited in their computational capabilities. These characteristics should be taken into account with regard to the design challenges.

1.2 Motivation

A common requirement of most applications in WSNs is a sensing system with a long-life. Unfortunately, according to a basic characteristic of WSNs, the sensors are powered by batteries, which are their sole energy source. There are a number of ways to harvest energy for rechargeable batteries, such as the vibration of rail tracks producing electromagnetic energy [13] [14], solar-powered systems [15], wind and water energy [16], but the sensors in these systems require deterministic and manual installation. Hence, it is impractical to recharge sensors’ batteries, particularly in the systems that the sensors are randomly dropped, perhaps, from airplanes or vehicles into target areas [17], which are in vast, remote and extreme environments e.g., forests, mountains and cultivated fields. Furthermore, in such random sensor deployment, the density of the deployed sensors is very high in order to compensate for the lack of exact positioning [18]. The network lifetime of a WSN may thus be shortened as a consequence of the high rate of networking overheads and the traffic load. These factors are also a cause of data collision due to the difficulty in management of media access control (MAC). Therefore, energy efficiency is crucial issue in WSNs.

There are many solutions that can avoid such issues by reducing amount of the data load, which is generated into the network, such as data prediction [19] and event-driven data reporting [20]. Indeed, sensor scheduling (or duty-cycling) is the one of the most effective solutions of significantly prolonging the network lifetime. A subset of sensors is dynamically active in order to carry out their tasks, whilst some others are in sleep mode to save energy. A
number of studies have been conducted in this area such as [21], [22] and [23]. However, traditional scheduling approaches have not paid enough attention to the quality of service in terms of sensing coverage and network connectivity, which may be impacted and degraded when the sensors are being scheduled.

The quality of both sensing coverage and network connectivity has to be guaranteed to be at the required level for all time regardless of how many sensors are active or asleep. Guaranteeing the quality of sensing coverage is important in order to ensure that all of the targets in the monitored field are covered by active sensors with a desired coverage degree. Meanwhile, guaranteeing the quality of network connectivity is important in order to ensure that the sensed data from each sensor can be delivered to a base station. Therefore, the principal motivation of this research is the desire to maximise the network lifetime using sensor scheduling and to maintain the sensing coverage and network connectivity at the required quality.

1.3 Research Objectives

The main objective of this research is to design sensor scheduling schemes that meet all of the following requirements:

- To guarantee the quality of sensing coverage and network connectivity: In stationary wireless sensor networks, the connected coverage can be guaranteed if and only if there is a sufficient density of randomly deployed sensors [24]. The challenge is how to guarantee that the connected coverage is of the required quality, when the sensors are scheduled. Our requirement is that the quality of sensing coverage is such that the entire monitored area must be fully \( k \)-covered, where \( k \) is a desired coverage degree (or \( k \) redundant sensors). Meanwhile, the quality of network connectivity is such that each active sensor must have at least one connection.

- To optimise coverage efficiency: Coverage efficiency is how an overlapping coverage degree produced by active sensors is minimised to be close to a desired coverage degree for the whole monitored area. The overlapping coverage degree is the quantity of actual intersection areas of sensing coverage that are over the desired coverage degree \( k \). Most studies in the literature guarantee the coverage quality at the desired coverage degree without considering the overlapping coverage degree [25] [26] [27].
In fact, this not only degrades the coverage efficiency but also wastes the sensors’ energy as it uses more active sensors than necessary.

- To balance energy consumption: Balancing energy consumption means that all of the sensors consume energy equally throughout the network’s lifetime. As the active sensors usually consume much more energy than those that are asleep, the network might incur the problem of a coverage hole as certain sensors deplete the energy. Therefore, balancing the energy consumption has to be considered in order to distribute the energy consumption equally among the sensors.

- To minimise the number of active sensors: Obviously, the number of active sensors for each scheduling round directly reflects the energy consumption of a WSN. The challenge is how to minimise the number of active sensors such that the quality of connected coverage is maintained at the required quality.

1.4 Main Contributions

In order to approach the research objectives mentioned above, four methods of sensor scheduling are proposed. The main contributions of the research in this thesis are as follows:

- A novel method of sensor scheduling using a virtual hexagon partition is proposed (Chapter 3). The virtual hexagon partition consists of a number of consecutive hexagonal cells as a hexagon tessellation that virtually covers a monitored field. Through a particular structure of the hexagon partition, the sensors deployed in the monitored field can obtain their position on the virtual partition in terms of a hexagonal cell and a triangular piece inside the cell into which they are grouped based on their geography (Section 3.2). In each operational round, $k$ sensors are selected to be active from each triangular piece of the hexagonal cells to fully $k$-cover the entire area of the monitored field (Section 3.4). As a hexagonal cell is composed of 6 triangles, the proposed method is called as the $6$-$Tri$ method.

The $6$-$Tri$ method gives perfect quality of connected coverage, which the entire area of the monitored field is fully $k$-covered and every active sensor is connected to another. This can also conserve the sensors’ energy by activating a certain number of sensors, such that the quality of connected coverage is maintained. Furthermore, the energy consumption is well balanced as every sensor has an equal chance of being
selected among the sensors and as they are regularly rotated to be on duty (Section 3.5).

- Another sensor scheduling method is proposed. Instead of the hexagon tessellation, this method uses a square tessellation as its virtual partition (Chapter 4). Similar to the previous method, the virtual square partition comprises a number of square cells. Each cell is composed of four sub-squares into which the sensors, which are randomly deployed in the monitored field, are grouped (Section 4.2). In each operational round, the sensor scheduling method activates $k$ sensors from each group of the sub-squares. Thus this method is named as the 4-Sqr method (Section 4.3).

The novelty of the 4-Sqr method is the utilisation of a virtual square partition to guarantee the quality of connected coverage while the sensors are being scheduled. The 4-Sqr method can retain all of the features achieved by the 6-Tri method. In particular, the number of active sensors activated by the 4-Sqr method is less than the number activated by the 6-Tri method (Section 4.4).

- Both methods – the 6-Tri and 4-Sqr – partition the monitored area into many cells of particular shapes – hexagon and square – respectively. The sensors are simply selected to be active according to the structure of the cells. Although the virtual square partition is better than the virtual hexagon partition in terms of the number of active sensors, the structure of the hexagon is more flexible and can be used to optimise the number of active sensors selected. The proof shows that the optimum number of active sensors is 3 for each hexagonal cell, which is only half the number for the 6-Tri method (Section 5.2).

- Based on the optimum solution proven in Section 5.2.3, a novel sensor scheduling method called the 3-Sym method is proposed. Three sensors are selected to be active from the dynamic symmetrical areas inside a hexagonal cell. There are two consecutive steps: the first is to select an independent sensor and the second is to select another two dependent sensors. In each cell, the sensor with the highest residual energy in the cell is selected as an independent sensor, and then another two sensors are selected within the symmetrical areas according to the independent sensor’s position.
The outstanding features of the $3-Sym$ method are that it can achieve not only complete connected coverage but also energy efficiency. The $3-Sym$ method conserves energy much more than either the $6-Tri$ or the $4-Sqr$ method. Moreover, energy is very well balanced among the sensors.

- An enhancement of the $3-Sym$ method is proposed that focuses on minimising the overlapping coverage degree; this is called the $O-Sym$ method (Section 5.4). The method has the $3-Sym$ algorithm as the sensor selection basis; however, it further optimises the overlapping coverage by checking the coverage redundancy. Every active sensor selected by the $3-Sym$ method investigates the coverage produced by others in its neighbourhood and decides whether or not it is eligible to sleep.

The $O-Sym$ method achieves all of the main research objectives. Moreover, the greatest accomplishment of the method is that the coverage efficiency is nearly ideal. In addition, the $O-Sym$ method has the longest network lifetime out of the methods.

1.5 Thesis Structure

The thesis is structured as follows:

Chapter 1 introduces a background of wireless sensor networks, including the architecture of a sensor node. Motivation is drawn to express limitations of wireless sensor networks that can significantly impact networks’ services. In addition, it points at a principal problem which will be addressed for this research. The objectives and main contributions are given respectively.

Chapter 2 describes the background, including the problem statements and the properties of sensing coverage and network connectivity, as well as strategies for connected coverage controls and sensor scheduling. This chapter also provides the state-of-the-art solutions and a number of recent studies are categorised and compared. In particular, the gaps are clearly identified and taken in to account in the rest chapters.

Chapter 3 proposes a sensor scheduling method using a virtual hexagon partition, called the $6-Tri$ method. A theorem along with its proof is given to ensure that the quality of connected coverage can be guaranteed all of the time while the sensors are being scheduled. An analysis
is introduced to evaluate the method’s performance and the results are validated by simulation. This work was published in [28].

Chapter 4 introduces another sensor scheduling method using a virtual square partition, called the 4-Sqr method. The structure of the square partition that is relevant to properties of sensing and communication of sensors is explained. The two phases of the method are detailed – position determination and sensor selection. The performance of the 4-Sqr method is evaluated and compared to that of the 6-Tri method. This study was published in [29].

Chapter 5 focuses on the optimisation of the number of active sensors and coverage efficiency. The optimum number of active sensors is introduced and proved. Based on the optimum number of active sensors, a novel method of sensor scheduling, namely the 3-Sym method, is proposed. Furthermore, an enhancement of the 3-Sym method called the O-Sym method, which optimises the overlapping coverage produced by the 3-Sym method in order to approach efficient coverage, is described. Simulation results show the performance of the O-Sym method compared to the previous methods and other existing schemes. This work was published in [30].

Chapter 6 concludes the thesis and gives a perspective on future works.
Chapter 2

Background and State of the Art

In this chapter, the background to quality of sensing coverage and network connectivity (known as the quality of connected coverage) is discussed. Sensor scheduling under the criterion of connected coverage assurance is also described. The existing state of the art approaches is categorised into groups based on the technique used. In particular, we compare and explain the advantages and drawbacks of the approaches.

2.1 The Connected Coverage Problems

Guaranteeing the quality of connected coverage is important in many applications. Each application has different types of targets for which the connected coverage needs to be guaranteed, such as points or borders, or even the entire area of the monitored field. This creates different scenarios and different problems as follows.

2.1.1 Discrete point connected coverage problem (DPCC)

Discrete point connected coverage (DPCC) is a surveillance problem where particular points or locations are monitored [18]. A set of discrete points of interest (DPOIs) is distributed within a monitored field. The main requirements of the problem are that the DPOIs are fully covered for all of the operational time and the sensed data can be delivered from each source node to the base station (BS). In such a randomly deployed sensor network, a DPOI is probably covered by more than one sensor or vice versa [31]. Therefore, the deployed sensors can be divided into subsets of covers, each of which can cover all of the DPOIs. Furthermore, as the DPOIs are discrete and distributed within the monitored field, a certain number of sensors may be additionally selected to act as relaying nodes to maintain the network connectivity. The subsets of covers will be recursively scheduled to operate for a certain period. There are many works in literature that describe this scenario e.g. [32].
Chapter 2. Background and State of the Art

Figure 2-1 illustrates an example of the DPCC problem in which four DPOIs are distributed in a monitored field. There are two disjoint subsets of sensors that can separately cover all of the DPOIs. Therefore, the network lifetime can be extended by scheduling these subsets to either operate or sleep for a certain period (shown in solid and dash circles respectively), whilst the quality of connectivity is maintained by relaying nodes. The DPCC problem can be defined as follows:

**The DPCC problem**

Given a monitored area $\mathcal{A}$, a set $S$ of sensors ($S = \{s_i | i = 1, 2, ..., N\}$), and a set $P$ of DPOIs ($P = \{p_m | m = 1, 2, ..., N_{DPOI}\}$), the DPCC is to find subsets of active sensors, each of which can cover all of the DPOIs in the monitored area $\mathcal{A}$ and can connect to a BS. The subsets of the active sensors operate for a certain interval and then are scheduled, which aims at maximising the network lifetime.

**2.1.2 Partial area connected coverage problem (PACC)**

Partial area connected coverage (PACC) is a surveillance problem that lies mainly in security applications such as intruder detection and border protection [33], known as barrier coverage [34]. In such a randomly deployed sensor network, a set of sensors is formed as a coverage barrier along a boundary of a security zone, rather than covering the whole area [35].
Chapter 2. Background and State of the Art

An intruder or mobile object trying to penetrate and cross at any point of the barrier can be precisely detected.

Kumar et al. [34] introduced $k$-barrier coverage for a given belt region. The given belt region, the dimension of which is expected to be long (as internal borders of a monitored area) and thin, is defined by two parallel boundaries with width $w$. Within the belt region, $k$-barrier coverage is formed as a detecting line for intruders, surrounding the monitored area. A crossing path is a route by which an intruder may cross from one parallel boundary to another boundary in the belt region. A set of sensors deployed over the belt region is considered to be

Figure 2-2: Partial (barrier) area coverage problem: (a) the network can provide 2-barrier coverage and (b) the network cannot provide barrier coverage (redrawn from [36])
the $k$-barrier coverage, if and only if all of the possible crossing paths of intruders are blocked by at least $k$ detecting lines ($k$ barriers).

In Figure 2-2, randomly deployed sensors in an open belt region can provide barrier coverage, depending on the following properties. Let $G(N) = (V, E)$ be a coverage graph, where $V$ is a set of sensors, $E$ is a set of sensing edges between a pair of the sensors, and $N$ is the number of sensors. An edge between two sensors exists if their sensing areas overlap each other within the belt region. In addition, there are two virtual sensors, source and destination, which correspond to the left and right boundaries of the belt region. A virtual edge between the sensors and the virtual source (or destination) exists if the sensing region of the sensors intersects with the left (or right) boundary of the belt region. The properties of the coverage graph provide the conditions for the existence of the $k$-barrier coverage [31].

For a network in which sensors are randomly deployed, the sensor scheduling can effectively conserve energy consumed by sensors performing any activity in security applications. Several sensor scheduling algorithms have been proposed to deal with the PACC problem such as [35], [36], [37] and [38]. Sensors randomly deployed in a belt region can be activated alternatively to form $k$-barrier coverage paths. The quality of the coverage and connectivity in a barrier coverage path must be satisfied. The PACC problem can be defined as follows:

<table>
<thead>
<tr>
<th>The PACC problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a belt region $\mathcal{R}$, and a set $S$ of sensors in the belt region ($S = {s_i</td>
</tr>
</tbody>
</table>
2.1.3 Entire area connected coverage problem (EACC)

Entire area connected coverage (EACC) is a problem that is encountered when vast areas such as forest fire monitoring [39] [40], environment monitoring [41] [42], and agricultural precision farming [43] [44] are continuously monitored. In a random sensor network, the number of deployed sensors is critical as this can directly affect the quality of connected coverage. Thus, sensors are randomly deployed with high density and are extensively distributed over the entire monitored area. Based on this scenario, the network is likely to face excessive collisions and redundant sensing if all of the sensors perform their tasks simultaneously (e.g., sensing and transmitting) [45]. Eventually, the sensors will gradually dissipate their energy and this will significantly reduce the network lifetime.

Therefore, sensor scheduling can be used to effectively address this issue. A certain number of deployed sensors are selected to be active as necessary, whilst the other sensors can be pushed into sleep mode in order to conserve their energy (as a sensor in sleep mode consumes much less energy than one in active mode). However, the sensor scheduling may degrade the quality of the connected coverage. Therefore, the quality of the connected coverage has to be maintained at the stratified level. Figure 2-3 depicts an example of the EACC problem where a subset of sensors is selected to be active. By employing a certain number of active sensors, every point in a finite area can be fully covered and each active sensor is connected to the others. The EACC problem can be defined as follows:

![Diagram of EACC problem](image-url)
Chapter 2. Background and State of the Art

2.2 Relevant Properties

Several properties of the elements in a network are highly relevant in considering the sensing coverage and the network connectivity. This section describes particular properties, including sensor properties, sensing coverage, network connectivity and the manner in which the algorithms are designed.

2.2.1 Sensor properties

Sensors have various properties, including capability, mobility and sensor deployment. These properties influence the algorithms designed to control the quality of the connected coverage.

2.2.1.1 Sensor’s capability

The capability of a sensor refers to the range of sensing coverage and communication as well as the capacity of energy resources. Sensors deployed in a network can be classified into one of two capability types i.e., homogeneous or heterogeneous. In the homogeneous type, all of the sensors have the same capabilities [46] [47] [48], while in the heterogeneous type sensors in the same network have different capabilities [49] [50] [51]. The sensor’s capabilities are usually assumed and defined as principal parameters in a network scenario.

2.2.1.2 Sensor’s mobility

The mobility of the sensors is a highly influential factor with regard to the quality of the connected coverage. Moving of a sensor from one place to another may cause a coverage hole to be incurred in the monitored area, as the sensing area produced by the sensor is finite. Similarly, sensors may be disconnected due to such a movement. Although many algorithms

---

The EACC problem

Given a monitored area $\mathcal{A}$, and a set $S$ of the deployed sensors ($S = \{s_i | i = 1, 2, ..., N\}$), the EACC is to find subsets of active sensors, each of which can $k$-cover every point in the monitored area and each active sensor must have at least $m$ connections to the others. Each subset of the active sensors operates for a certain interval, and then is scheduled to another subset, which aims at maximising the network lifetime.
have been proposed under a stationary scenario, the mobility of the sensors is essential in some scenarios. For example, sensors are randomly deployed in an unknown and hostile environment in which they need to relocate [52] [53] [54]. Thus, algorithms designed under sensor’s mobility are much more complicated than those designed in static networks.

2.2.1.3 Sensor’s deployment

Node deployment is a fundamental issue that significantly influences the control of the connected coverage. Sensors can be deployed in either a determined or a random manner [55]. The former is suitable for known areas; on the other hand, the latter is suitable when the sensors are deployed with high density and when large numbers of sensors are used in particularly inaccessible areas.

2.2.2 Sensing coverage properties

The sensing coverage properties represent sensors’ measuring capability. In general, a sensor can effectively sense and capture circumstances in a particular manner. As different applications require different manners [56], this section describes several aspects of the sensing coverage properties, including models, shapes and coverage degrees.

2.2.2.1 Sensing coverage models

A sensing coverage model represents an operational space, in terms of distances and angles, in which a sensor can effectively sense and capture any phenomena. In particular, it indicates the quality of the sensing coverage over targets through the associated angles and distances between the sensors’ and targets’ locations.

Let \( s_i \) be a sensor in a network for \( i = \{1, 2, 3, ..., N\} \). A sensor’s location and a point’s location are denoted as \((x_i, y_i)\) and \((x, y)\), respectively. The Euclidean distance between sensor \( s_i \) and point \( q \) is given by \( d_{s_i, q} = \sqrt{(x_i - x)^2 + (y_i - y)^2} \). There are three different coverage models as follows:
Chapter 2. Background and State of the Art

Binary disc coverage model: The binary disc is the simplest coverage model. It has been widely used in many works in the literature [31], for example [57]. The function of the binary disc coverage model is given by:

\[ c(s_i, q) = \begin{cases} 
1, & \text{if } d(s_i, q) \leq R_s, \\
0, & \text{otherwise}, 
\end{cases} \tag{2-1} \]

where \( R_s \) is the sensing radius. Obviously, this defines the sensing area of a sensor, centred at the sensor’s location with the sensing radius. Figure 2-4(a) depicts the sensing area of a sensor with the sensing radius \( R_s \). According to (2-1), any single point in the monitored field is covered by a sensor if it is within the sensor’s sensing area \( d(s_i, q) \leq R_s \). The quality of coverage is equal for all points within this space. On the other hand, any single point outside the sensor’s sensing area is not covered.

Probabilistic coverage model: The probabilistic coverage model is more realistic than the binary disc coverage model (see Figure 2-4(b)). Typically, the detection capability of sensing devices varies according to the distance from a target, for example ultrasound and radar [56] [58]. Thus, the model can be formulated as:

\[ c(s_i, q) = \begin{cases} 
1, & \text{if } d(s_i, q) \leq R_s - R_u, \\
e^{-\lambda d^\beta}, & \text{if } R_s - R_u \leq d(s_i, q) \leq R_s, \\
0, & \text{if } d(s_i, q) > R_s, 
\end{cases} \tag{2-2} \]
where $R_u$ is called the uncertain range, $\lambda$ and $\beta$ are constants, and $d = d_{s_i,q} - (R_s - R_u)$. The quality of the sensing coverage can be defined as a detection probability function. The detection probability within an uncertain range decreases exponentially when the distance from the target increases. It eventually reaches zero when the target is beyond the sensing radius. Figure 2-5 illustrates the variation in detection probability for sensing devices with different values of $\lambda$ and $\beta$. Since this sensing coverage model is realistic, many studies have used this model, for example, [59] and [60].

**Binary sector coverage model:** Some types of sensing devices may operate for a particular direction and angle, such as infrared and cameras [61] [62]. The binary sector coverage model can truly reflect the sensing coverage produced by such devices (see Figure 2-4(c)). Many studies have used this model [63] [64] [65] [66]. The binary sector coverage model is given by:

$$c(s_i, q) = \begin{cases} 1, & \text{if } d(s_i, q) \leq R_s \text{ and } \phi_s \leq \phi(s_i, q) \leq \phi_s + \omega, \\ 0, & \text{otherwise}, \end{cases}$$

(2-3)

where $\phi_s$ is the orientated angle and $\omega$ is the sensing angle. $\phi(s_i, q)$ is the angle measured from a horizontal line ($0^\circ$) to a target point $q$, centred at a sensor $s_i$. According to (2-3), any single point within the sector coverage area of a sensor $s_i$ is considered to be covered by the sensor; any single point outside the sector area is considered not to be covered by this sensor.
2.2.2.2 Sensing coverage degrees

The sensing coverage degree indicates coverage redundancy over a single point in a monitored field. Let $\mathcal{D}(s_i, R_s)$ be a sensing coverage area of a sensor $s_i$ with a sensing radius $R_s$. Typically, the sensing coverage area could be determined by either the binary disc or sector coverage model. As mentioned above, any point $q$ is covered by $s_i$ if $q \in \mathcal{D}(s_i, R_s)$. In addition, the point $q$ is $\mu$-covered if it is in the intersection area of the sensing areas of $\mu$ sensors, $q \in \bigcap_{i=1}^{\mu} \mathcal{D}(s_i, R_s)$, where $\mu$ is the actual coverage degree produced by the sensors.

Some applications may require redundant coverage in order to tolerate sensor failure. Thus, the quality will be guaranteed at the desired coverage degree ($k$). Figure 2-6 illustrates different coverage degrees i.e., {1, 2, 3 and 4} in particular areas produced by 4 sensors for both the disc and sector coverage models. If the desired coverage degree, for example, is $k = 2$, then the quality of $k$-coverage upon the particular areas with 1 degree is not satisfied.

For the probabilistic coverage model, the coverage, which is made by a number of sensors, cannot be considered as the degree, but is instead considered as the desired probability of coverage. The coverage probability is given by:

$$C(q) = 1 - \prod_{i \in S} (1 - c(s_i, q)) > \varepsilon,$$

(2-4)
where $S$ is a set of deployed sensors and $\epsilon$ is the predetermined threshold by the application [67].

2.2.3 Connectivity properties

The properties of connectivity are expressed with regard to a link connection between a pair of sensors rather than a networking connection. Connectivity is another important factor that significantly impacts the network’s performance. Thus, its quality must be maintained at the satisfied level so that it is similar to the quality of the coverage. Nevertheless, controlling the connectivity is likely to be simpler than controlling the quality of coverage as only available neighbours need to be considered [68]. As the properties of connectivity are closely associated with the coverage properties, the relationship between them is also described in this section.

2.2.3.1 Communication models

**Binary disc communication model:** Similar to the binary disc coverage model, the binary disc communication model is the simplest model in which the communication ability is based solely on the Euclidean distance between sensors. Let $R_c$ be the communication radius. The connectivity is such that the sensors $s_i$ and $s_j$ are connected if the Euclidean distance between them is not greater than the communication radius, i.e.

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_c.$$  

If every sensor has a connection link to another, then its sensed data can be delivered throughout the network towards the BS.

In the case of a heterogeneous network, the sensors may have different communication radii, i.e., $R_{c_i} \neq R_{c_j}$. Thus, two sensors $s_i$ and $s_j$ are able to communicate with each other if the distance between them is not greater than the minimum radius of the communication themselves, i.e., $d_{i,j} \leq \min\{R_{c_i}, R_{c_j}\}$ [45]. Note that the connectivity means the bidirectional communication.

**SINR communication model:** SINR stands for the signal-to-interference-plus-noise ratio. In reality, there are crucial factors that degrade the quality of the signal on the receiver side in wireless communication, i.e., noise, path loss and interference [69]. Noise, which exists in the environment, such as thermal noise and cosmic noise, is a white Gaussian random process that is added prior to the transmitted signal reaching the receiver [70]. The power of the
received signal \((P_r)\) is basically attenuated over a distance due to path loss. A simple model of the path loss can be given by

\[
P_r = \frac{P_t}{(d_{i,j})^\sigma}
\]  

(2-5)

where \(P_t\) is the transmitted power, \(P_r\) is the received power and \(\sigma\) is the path loss exponent. Furthermore, the received signal may be interfered by unwanted signal from unintended sensors. Therefore, the communication is successful if the received signal is at least a certain threshold \((SINR_0)\) as:

\[
\frac{P_r(s_i)}{f + \sum_{s_k \in \mathcal{N} \backslash s_i} P_r(s_k)} \geq SINR_0,
\]  

(2-6)

where \(P_r(s_i)\) is the signal power received at a sensor \(s_j\) from a sensor \(s_i\), \(P_r(s_k)\) is the signal power received at a sensor \(s_k\), where \(s_k \in \mathcal{N} \backslash s_i\), from a set of its neighbours except the sensor \(s_i\), and \(f\) is the noise.

### 2.2.3.2 Relationship between communication and sensing radii

To control the quality of connectivity and coverage, the relationship between them needs to be investigated. Typically, the communication radius to the sensing radius can be

![Diagram of communication and sensing radii](image-url)

Figure 2-7: Relationship between the communication and sensing radii

(a) \(R_c/R_s > 1\), (b) \(R_c/R_s < 1\) and (c) \(R_c/R_s = 1\)
\[ R_c/R_s > 1, \quad R_c/R_s < 1 \quad \text{and} \quad R_c/R_s = 1 \] as shown in Figure 2-7(a), (b) and (c), respectively.

The authors in [68] [71] have proved that the best quality of the connected coverage can be achieved if the communication radius is greater than twice of the sensing radius \( R_c/R_s \geq 2 \).

In addition, many works in the literature have taken this as their assumption [72] [73]. However, this assumption does not reflect the reality because the communication ability and sensing of the sensors can be of many kinds as mentioned above. Therefore, some other proposed works have been designed based on any kind of relationship, especially in heterogeneous networks [74].

### 2.2.3.3 Connectivity degree

The degree of connectivity is the number of connection links to different neighbours of a sensor. A sensor with only one connection may be a bottleneck in a network. The bottleneck is a critical cause of the network to be partitioned apart. Figure 2-8 depicts the bottleneck at node 3 to which nodes 1 and 2 have a single connection. The failure of node 3 potentially partitions both nodes 1 and 2 apart from the rest of the network. Therefore, some applications require that a sensor must have \( l \) degrees for connectivity redundancy [75].

### 2.2.4 Algorithm properties

The algorithms regarding controlling the connected coverage can be classified into two groups: distributed [76] and centralised [77]. In the former, every sensor performs
individually and decides for itself whether to be active or sleep. Certain information, such as a sensor’s location and residual energy, may be exchanged locally among neighbours. For the latter, the process and any decision are mainly performed at the BS’s side. In this case, the relevant information from the sensors is collected across the network to the BS.

2.3 Connected Coverage Verification and Control

This section describes how to verify and control the quality of connected coverage within a given local area. There are several strategies that can be used to solve objectives in different kinds of problems, including the discrete point coverage (DPCC), the partial area connected coverage (PACC) and the entire area connected coverage (EACC). These strategies are basically adopted as a fundamental principle in algorithms for controlling the quality of the connected coverage on a global scale. To describe this clearly, the strategies are distinguished into coverage control and connectivity control.

2.3.1 Coverage control strategies

The aim of coverage control strategies is to ensure that either every target point or the entire target area is covered at the required quality in terms of the desired degree \((k\text{-cover})\) or the coverage probability. To simplify the following description, the Boolean disc is used as a coverage model and \(k = 1\). In controlling the quality of coverage on target points, the DPCC problem is much easier to overcome than the target areas i.e., the PACC and EACC problems. The quality of coverage on different types of targets can be verified and controlled using the following strategies:

**Considering Euclidean distance:** Using Euclidean distance is a straightforward strategy that can indicate the quality of coverage produced by a sensor on a target point, especially with regard to the DPCC problem. As with any coverage model mentioned in Section 2.2, the ability of sensors to cover targets has distance as a function. Therefore, the quality of coverage on a target point can be verified by considering the Euclidean distance between a sensor and the target compared to the sensing range of the sensor.
**Considering proportion:** Solely considering the distance to the points is not sufficient to qualify the coverage in a particular area, which is a target in the PACC and EACC problems, because there can be an immeasurable number of point spaces in a finite area. Thus, the simplest way is to consider proportion of the covered area to the total monitored area as

\[
\Pi(\mathcal{A}) = \frac{\bigcup_i \mathcal{D}(s_i, R_s)}{\mathcal{A}},
\]

(2-7)

where \(\Pi(\mathcal{A})\) is the proportion of the covered area to the total area \(\mathcal{A}\), and \(\bigcup_i \mathcal{D}(s_i, R_s)\) is the...
area covered by all of the sensors $s_i$ with sensing radius $R_s$. To determine the covered area, each sensor may count grid points, which are marked across the monitored area, within its sensing area [31], as shown in Figure 2-9. However, it is impractical to mark such a grid of points and count it.

**Considering intersection point:** To be more practical and self-organised, Xing et al proposed an algorithm to determine the coverage quality by considering intersection points [78]. The intersection points are produced when the sensing discs of two sensors intersect each other. An intersection point $q_{ij}$, which is produced by two associated sensors $s_i$ and $s_j$, is considered to be covered if it is in the sensing area of another sensor $s_l$. The sensing area of sensor $s_i$ is fully covered if all of the intersection points produced within this area are covered by other than sensor $s_i$ itself. Figure 2-10 depicts an example of both covered intersection points and uncovered intersection points. The intersection point $q_{12}$, produced by sensor $s_1$ and $s_2$, is not covered by any sensor, but sensor $s_5$. Therefore, sensor $s_5$ is ineligible to enter the sleep mode. Some studies used this strategy [79] [80] [78].

**Considering perimeter coverage:** The coverage on a sensing area’s perimeter (simply called the perimeter of a sensor) can also be used as an indicator of its quality [81] [82]. When a sensor $s_i$ is intersected by its neighbour $s_j$, a coverage arch on the sensor’s perimeter is produced, where $\alpha_{(s_j,L)}$ is the arch’s beginning angle and $\alpha_{(s_j,R)}$ is the arch’s ending angle.

![Diagram showing coverage arches and perimeter coverage](image-url)

Figure 2-11: Illustration of perimeter coverage (a) coverage arches on sensing perimeter produced by intersection between sensing areas (b) the sensor’s perimeter is under the coverage arches (redrawn from [83])
Any single point on the sensor’s perimeter under the coverage arch is covered by the
neighbour. The sensing area of sensor $s_i$ is fully covered if all of its perimeter and its
neighbours’ perimeters within the sensing area are under the coverage arches produced by
other sensors than $s_i$. Figure 2-11 illustrates that the sensor $s_3$ is satisfied with the condition,
and it is therefore eligible to enter the sleep mode.

### 2.3.2 Connectivity control strategies

The aim of connectivity control strategies is to ensure that every active sensor can
communicate with the base station (BS), either by direct communication or multi-hop
communication. To simplify explanation of connectivity control strategies, the binary disc
communication model is used to represent the connectivity property. Connectivity control
strategies can be classified into three categories as follows:

**Sensing coverage implying connectivity:** The sensing coverage and the connectivity have a
significant relationship as mentioned in Section 2.2.3. If the relationship between the sensing
and communication radii as $R_c \geq 2R_s$ is satisfied, then the quality of the coverage obviously
implies the quality of the connectivity [71]. Therefore, the coverage at the required quality
can be solely maintained and, in return, the quality of the connectivity can be guaranteed. In
Figure 2-12, the maximum distance between sensor $s_i$ and $s_j$ is $d_{ij} = 2R_s$ such that the
coverage is acceptable (no gap). Obviously, the connectivity of these sensors is satisfied

![Figure 2-12: Coverage implying connectivity by relationship $R_c \geq 2R_s$](image-url)
because $R_c = d_{ij} = 2R_s$. However, this strategy is valid only in homogeneous networks [83].

**Additional relaying sensors required:** In some scenarios e.g. the DPCC, the number of active sensors covering targets is insufficient to connect and deliver their sensed data to the BS due to a few number of them. In this case, additional sensors (in addition to the active sensors for covering the targets) are required, in order to relay the sensed data from the covering sensors to the BS, especially in multi-hop communication networks [74].

**Independent connectivity control:** Although a certain number of sensors are activated across a network to maintain complete coverage, the quality of the connectivity cannot be guaranteed or implied by the coverage being maintained, especially in heterogeneous networks. In this kind of network, the sensors’ capabilities are different, and the relationship between the sensing and communication radii is uncertain. Therefore, the connectivity problem must be taken into account separately. Some of these strategies can be found in [84].

### 2.4 Sensor Scheduling

As sensors typically have limited energy resources e.g., batteries [21] [85], energy conservation is important in order to prolong the life of the network. Sensor duty scheduling is a way to effectively conserve the energy consumption of the sensors [56]. Since a sensor undertaking different duties consumes different amounts of energy, sensor scheduling (or duty cycling) can minimise and balance the energy consumption by assigning the sensors proper duties [86] [87].

Sensors carrying out their duties can be in either active or sleep mode. In active mode, all of the components, such as the sensing unit and transceiver are in operation. In this mode, a sensor is able to sense and transmit its data as well as receive data transmitted from its neighbours in order to forward the data to the BS. Sensors in this mode therefore have a high rate of energy consumption even if they are in an idle state and just listening, as shows in Table 2-1 [23]. On the other hand, sensors in sleep mode turn most of their components off except for a triggering mechanism e.g., a timer to wake them up.
There are two methods of sensor scheduling:

**Round of certain duration:** The network lifetime is divided into rounds of a certain operational duration [88] [89]. In each round, a certain number of sensors are active in order to maintain the quality of the connected coverage, while the other sensors are pushed into sleep mode. The active sensors operate for a while, and then all of the sensors are rotated for the next round in order to balance the energy consumption among them.

In general, there are two phases in each round, i.e. sensor configuration and operation. In the former, the sensors decide whether to be active or not, and in the latter, the active sensors perform their responsibilities. The operational duration is usually much greater than the configuration duration as shows in Figure 2-13. Certainly, synchronisation among the sensors is required for this scheduling aspect. This can be done at the beginning of each round.

**In-time Checking:** Unlike the operational round scheduling, the in-time checking mechanism allows the sensors that are currently active to carry out their duties continuously until their

---

Table 2-1: Energy Consumption Values of Sensor in Different Duties [23]

<table>
<thead>
<tr>
<th>Energy consumed</th>
<th>Typical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy consumption in transmitting</td>
<td>60 mW</td>
</tr>
<tr>
<td>Energy consumption in receiving</td>
<td>45 mW</td>
</tr>
<tr>
<td>Energy consumption in idle listening</td>
<td>45 mW</td>
</tr>
<tr>
<td>Energy consumption in sleeping</td>
<td>90 μW</td>
</tr>
</tbody>
</table>

---

Figure 2-13: Sensor scheduling by certain duration of the operational round
energy is exhausted [90]. The sleep sensors need to wake up regularly in order to check the associated active sensors’ status. Once they have found an abnormal active sensor, the sleeping sensors change their mode and replace it, such that the quality of the connected coverage is continuously maintained. In this way, the fault tolerance of the network is excellent as a faulty active sensor can be addressed within a short time. However, the energy consumption among the sensors is likely to be imbalanced due to the lack of a duty rotation.

2.5 Evaluation metrics

The approaches to guaranteeing connected coverage in sensor scheduling can be evaluated using several metrics. Thus the metrics are grouped into five aspects as follows.

2.5.1 Complexities of algorithms

The complexities reveal how the proposed algorithms are scalable. They are usually a function composed of the number of deployed sensors. There are two indicators in this aspect i.e., time complexity and message complexity.

Time complexity (TC): The time complexity indicates the computational time needed to solve the problem. An algorithm with a long computational time is likely to be more complicated.

Message complexity (MC): The message complexity represents the overhead of the controlling messages generated by an algorithm. The number of controlling messages can directly impact a sensor’s energy and also the data collision.

2.5.2 Constraints of algorithms

The constraints indicate the flexibility of the algorithm. Certain algorithms might require some assumptions in order to work and may be impractical. Therefore, to reflect the reality, algorithms should be free from any restricted assumptions and therefore more flexible. The following metrics are often found in the literature as constraints of algorithms.

Location (LC): A sensor’s location is important information that is usually required by algorithms to control the quality of the coverage. Although there are many ways to obtain such information, an additional component e.g., a global positioning system (GPS) or a positioning algorithm is required for each sensor. Therefore, it is much better that approaches are free of this requirement.
Sensor capability (SC): As mentioned in Section 2.2.1, sensors deployed in a network may have either the same capability (homogeneous) or different capabilities (heterogeneous). In fact, various types of surveillance are possible in a monitored field; thus different kinds and capabilities of sensors are required. Therefore, approaches based on heterogeneous sensor deployment are more flexible.

Communication-Sensing relationship (CSR): Similar reason with the sensor capability, the assumption of relationship between communication and sensing radii, \( R_c \geq 2R_s \), is useful and benefits to easily control the quality of the connected coverage as stated in Section 2.2.3. However, this relationship is quite impractical and too restricted. It is much more realistic to relax this consumption.

Synchronisation (SYN): Time synchronisation is necessary when sensors are scheduled using rounds of a particular duration. In each round, all of the sensors are synchronised with each other at the beginning and are then rotated for their duties in the configuration phase (see Section 2.4). However, synchronisation is not easy, particularly in wireless sensor networks. Unsynchronised timing among sensors yields some missing configuration.

2.5.3 Quality of the connected coverage

The quality of the connected coverage is the major aspect that is needed to evaluate the algorithm’s performance. Two terms need to be measured in this aspect i.e., the quality of connectivity and the quality of coverage. Although the detail of the connectivity and coverage was described in Section 2.2, here their quality is defined.

Quality of the coverage (QCV): The quality of the coverage can be estimated by investigating how the targets will be covered with a coverage degree \( k \). It is considered that the requirement is met if all of the targets are \( k \)-covered. Hence, the quality of the coverage is given by the proportion of \( k \)-covered targets (areas or points) to all of the targets.

Quality of the connectivity (QCN): Similar to measuring the coverage, the quality of the connectivity can be estimated by investigating how many connection links each active sensor has to its neighbours. It is considered that the requirement is met if every active sensor has at least \( l \) connectivity degrees. Thus, this metric is given by the proportion of the number of active sensors that have the required connectivity degree to the total number of deployed sensors.
2.5.4 Energy efficiency

Energy efficiency is another important issue in wireless sensor networks. There are two components that precisely represent the energy efficiency i.e., energy conservation and energy balance.

**Energy conservation (EC):** The network lifetime is a straightforward factor that shows how energy can be conserved. Another factor is the number of active sensors, which directly impacts the amount of energy that the system consumes. In active mode, sensors perform various tasks, so they consume much higher energy than those in sleep mode, according to Table 2-1. Approaches that activate a few sensors such that all of the requirements are met can certainly conserve more energy than ones with a higher number of active sensors.

**Energy balance (EB):** Balancing energy consumption among sensors is important for wireless sensor networks in order to stabilise their services. If they are consuming energy unequally, certain sensors are likely to die earlier. This could cause a bottleneck or even could partition a network apart as mentioned in Section 2.2.3.

2.5.5 Fault tolerance

Expectations that a network will maintain good service may be very high in some applications, particularly in military applications. Although a lack of full coverage is a small part, the consequences could mean big trouble for a security system, for example, the barrier coverage (see Section 2.1.2) in intrusion detection. Therefore, fault tolerance is important. Two metrics – sensor failure detection and location error tolerance – reflect this ability of approaches.

**Sensor failure avoidance (FA):** The failure of a sensor can cause a coverage hole and even a network partition. A load occurs when active sensors perform their duties such as sensing, transmitting and even idle listening. As these tasks mainly consume the energy of active sensors, a coverage hole may occur if certain sensors have a heavier load and rapidly deplete their energy [91]. Thus, it is necessary for the sensors’ load to be balanced. Load balance can be achieved by duty-cycling in order to evenly distribute the energy consumption among the sensors. Eventually, this can maximise the stability of a WSN. Furthermore, sensor devices are likely to become quickly dysfunctional for many reasons, such as hardware damage and energy depletion. A mechanism for checking sensor failure is necessary, not only for awareness, but also to heal any coverage holes.
**Location error tolerance (LET):** A failure in both coverage and connection may be a consequence of a sensor’s location error. As sensors obtain their position through a positioning process or GPS, there may be an error in the reading. Hence, an algorithm may significantly activate sensors in the wrong places, which can cause poor quality of connected coverage. This failure can be avoided or mitigated if approaches can tolerate the location error.

### 2.6 The State of the Art

State-of-the-art approaches in connected coverage assurances for the sensor scheduling is discussed. The approaches are grouped based on the techniques used. Furthermore, their advantages and drawbacks are explained.

#### 2.6.1 Geographical based strategy

In the geographical based strategy, information such as the positions of the sensors and targets is usually taken into consideration with regard to the coverage quality. This strategy focuses on considering the geographical circumstances of coverage due to the sensors over the targets in the monitored fields. The distances between the sensors and targets, as well as the sensing coverage models and coverage shapes are investigated to examine the quality of connected coverage.

##### 2.6.1.1 Geographical based approaches

**Coverage Configuration Protocol (CCP):** The CCP protocol investigates the coverage redundancy incurred within the sensing area of a sensor in order to decide whether this sensor is eligible to sleep [78]. To check the coverage redundancy, it exploits the intersection point strategy. There are three states in which each sensor can be – sleep, listen and active. The sensors are initially active and consider their eligibility. If a sensor finds that the coverage degree within its sensing area exceeds the requirement then it decides to enter the sleep mode. Sleeping sensors periodically wake up and enter the listening mode in order re-evaluate their eligibility, as a currently active sensor can run out of energy.

**Optimal Geographical Density Control (OGDC):** To minimise coverage redundancy and select a lower number of active sensors, the OGDC algorithm investigates intersection points produced by a pair of sensing discs [71]. According to Figure 2-14, point $O$, which is an intersection point between the sensing discs of sensors $s_i$ and $s_j$, is the critical point that
needs to be covered by another sensor. The determined point P is the best location for a sensor to be able to perfectly cover the critical point. The point P is used as a target to select another active sensor. However, finding a sensor in the exact position is impossible, particularly in a random network, and therefore the closest sensor will be selected instead. A timer is utilised to select the closest sensor as follows:

\[
T = t_0 \left( c \left( (R_s - d(O, s_k))^2 + (d(O, s_k) \cdot \emptyset)^2 \right) + u \right),
\]

where \( t_0 \) is the given time, \( c \) is a constant, \( \emptyset \) is the angle between the sensor’s position and the position of point \( P \), and \( u \) is a random value \([0,1]\).

**Demand-Based Coverage and Connectivity-Preserving Routing Protocol (DBCCR):** The main purpose of the DBCCR protocol is to reduce the energy consumption by minimising the adjustable sensing range of the sensors according to the desired coverage ratio \([88]\). There are three main phases in the CBCCR i.e. sensing radius setting, sensor selection and sensor connection. In the sensing radius setting phase, every sensor is assigned a sensing radius according to the expected desired coverage ratio demanded by the application. The expected coverage ratio, which was introduced by \([92]\), has a sensing radius as shows below:

\[
E[\Omega] = 1 - e^{-\rho \pi R_s^2},
\]

(2-9)
where $\Omega$ and $\rho$ are the coverage ratio and the network density, respectively.

After the sensing radius has been set by the BS, sensors are selected based on local information from the one-hop neighbours. The information used in the sensor selection phase is composed of three factors i.e. the residual energy, the utility, and the stability of each sensor. The utility, which is the number of neighbours, indicates the communication ability of the sensors and the stability expresses the quality of the links. Only sensor with the highest value for these factors will be active within its one-hop region. Note that after finishing this phase, none of the selected sensors have any connections among them yet. Hence, another sensor in a communication intersection area of the selected sensors will be activated in the sensor connection phase.

**Resilient Online Coverage (ROC):** The authors proposed a scheme called the Resilient Online Coverage (ROC) [93] to deal with the failure of sensors during an operational period in an energy-efficient manner. To be active, sensors can be selected using any approach among the Deterministic Cover Selection (DCS) algorithms e.g., [78] and [94]. Furthermore, the ROC allows the sensors to be scheduled using a technique called the Offline Cover Update (OCU). Using the OCU, sensors that are currently active as the proxy nodes of the currently sleeping sensors select a new set of active sensors for the next operational round.

In order to heal a coverage hole incurred due to a sensor’s failure, the sensing area of each active sensor is monitored by sets of backup sensors which are a certain number of the sleeping neighbours. The backup nodes will regularly probe their associated active sensors’ condition in certain intervals. Each of the backup sets must have the ability to fully cover the entire sensing area of an active sensor that is being monitored. To form the backup sets, a greedy algorithm is proposed. An active sensor greedily selects its potential neighbours based on two criteria i.e. the largest percentage of a common sensing area – an intersection area between the active sensor and its neighbour – and the highest number of sensors to be backup.

Xing et al, in [78], investigated the intersection points occurring between sensing discs of sensors. They claim that intersection points exist in the monitored area and that if every intersection point is $k$-covered, then the monitored area is fully $k$-covered.
2.6.1.2 Discussion

The CCP algorithm reveals an excellent quality of connected coverage because coverage redundancy is investigated for every single sensing area. In addition, a coverage hole can be rapidly healed as the circumstances are periodically checked by the sleeping sensors, so the algorithm can tolerate sensor’s failure. However, the CCP algorithm needs to verify the eligibility of each sensor with regard to whether it should be active. Every intersection point within a sensor’s sensing area is investigated. The CCP algorithm therefore takes time for \( O(N^3) \) to verify every intersection point produced by all of the neighbours surrounding a sensor. This amount of time complexity is considered that the CCP algorithm is slow to get a set of active sensors. In addition, the energy consumption is not well balanced as the active sensors must carry out their roles until their energy is depleted; and they will then be replaced by others. This can cause the network to be isolated at some critical points e.g., a bottleneck.

The OGDC algorithm has very low overheads. Its message complexity is only \( O(N_a) \), where \( N_a \) is the number of active sensors. In addition, the time complexity is also \( O(N_i^2) \), where \( N_i^2 \) is the number of neighbours of sensor \( s_i \). This means that the algorithm is scalable. However, the algorithm may produce several target points close to each other. Thus, this may mean that some of the active sensors, which are selected according to those target points, are too close.

Figure 2-15: The target points, \( P_1, P_2 \) and \( P_3 \) being close to each other
Figure 2-15 illustrates an example in which the target points \((P_1, P_2, \text{and } P_3)\) are produced by different pairs of the sensors. Obviously, any sensors to be selected according to these target points will be close to each other, and consequently the degree of coverage redundancy will be high.

The scalability of the DBCCR protocol is excellent as it has low overheads. These are \(O(N + N_a)\) and constant \(O(1)\) to the control messages and time complexities, respectively. In addition to considering the residual energy, the protocol selects sensors according to the link-failure rate, so that energy can be conserved by transmitting data without failure. As the coverage ratio is desirable based on the application’s demand, no controlling mechanism is able to identify the particular area of interest, as a target can be a part of the whole monitored field. Furthermore, the quality of connected coverage cannot be guaranteed by this protocol, as the one-hop announcement by the selected sensors cannot prove that the target areas are fully covered.

The novel features of the ROC approach are using the OCD technique to schedule sensors and heal coverage holes. In the former, the overhead produced for the sensor selection is low as only the currently active sensors, which are the proxies of other sleeping sensors, are involved in the process. For the latter, using an opportunistic approach to probe a sensor’s failure, the ROC can heal a cover-hole almost in real-time because the probe interval is estimated based on the node failure probability. Furthermore, the time complexity is \(O(N_t^2)\), which is similar to the best scale of the approaches in the DCS category. Besides, the message complexity is \(O\left(\frac{T}{t_s}(N + N_a) + \frac{T}{t_p}N_t\right)\). Although the scale is linear, the overhead of the controlling messages is quite high due to the frequency of the messages that are periodically generated to update topologies \(\left(\frac{T}{t_s}\right)\) and probe failure circumstances \(\left(\frac{T}{t_p}\right)\).

However, the ROC is excellent in terms of the constraints, as only the location information is required. Using the DCS method to select sensors, the ROC gives excellent quality of connected coverage inherited from the DCS. Indeed, the outstanding feature of this approach is the fault tolerance. The condition of every active sensor is monitored by a set of backup sensors.

However, the energy use is not efficient because of the fact that additional sensors are used as backup nodes as they need to generate a probing message periodically. In addition, a proxy might consume a huge amount of energy and might be too complex to consider other sleeping
sensors to be active for the next round. Therefore, this results in unbalanced energy. In healing a coverage hole, the ROC is likely to replace a failed node with an unnecessary number of compensated nodes. This is because it uses a set of backup nodes to replace and fully cover the entire sensing area of a currently active sensor rather than the exact size of the coverage hole.

Table 2-2: Achievements and Drawbacks of the Geographical based Approaches

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Achievement(s)</th>
<th>Drawback(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCP [78]</td>
<td>- High quality of connected coverage</td>
<td>- High time complexity</td>
</tr>
<tr>
<td>OGDC [71]</td>
<td>- Low overhead</td>
<td>- High rate of coverage redundancy</td>
</tr>
<tr>
<td>DBCCR [88]</td>
<td>- Demand-based coverage ratio is to reduce the energy consumption.</td>
<td>- Targets of interest, which can be a partial area, cannot be exactly identified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The quality of connected coverage cannot be guaranteed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Extra sensors are needed to relay sensed data.</td>
</tr>
<tr>
<td>ROC [93]</td>
<td>- Scheduling sensor by an offline cover update</td>
<td>- Overload on the proxy nodes to perform the sensor selection</td>
</tr>
<tr>
<td></td>
<td>- Almost real-time healing of coverage-holes incurred due to sensor failure</td>
<td>- Using a number of compensated nodes more than necessary to heal a coverage hole</td>
</tr>
</tbody>
</table>

2.6.2 Heuristic based strategy

2.6.2.1 Heuristic based approaches

Communication Weighted Greedy Cover (CWGC): The authors in [18] studied the Maximum Cover Tree (MCT) problem, and proved that it is NP-Complete. In this problem, all of the DPOIs must be covered by a certain number of sensing sensors and these sensors either connect directly or through relaying nodes to the sink (BS) as a tree topology. A set of sensing and relaying nodes are selected to undertake their roles for a finite period. Note that a relaying node can be one of the sensing nodes (non-disjoint set). The MCT problem is formulated as a Linear Program (LP) that aims to maximise the network lifetime. Base on the formulation of the MCT problem, a heuristic algorithm called CWGC has been proposed.

The CWGC uses a greedy heuristic method to construct sets of cover trees, each of which can cover all of the DPOIs and deliver all of the sensed data to the BS. The algorithm greedily selects source nodes – performing the sensing task over target points – by considering the maximum profit value of sensor $s_i$, which is given by the proportion between the number of uncovered target points and the path weight as:
where $|P_i - P'_i|$ is the number of target points within the sensing area of sensor $s_i$, which is not covered by any other sensors, and $w_{i,BS}$ is the path weight, which is the sum of the link weights from sensor $s_i$ to the BS. The link weight ($w_{ij}$) between a pair of adjacent sensors is given by:

$$w_{ij} = e_{ij}^t \times \frac{E_i}{e_i}$$

(2-11)

where $e_{ij}^t$ is the energy consumed by sender $i$ to transmit a bit of data to receiver $j$, and $E_i$ and $e_i$ are the initial and residual energies respectively.

In addition, the CWGC greedily constructs a set of minimum weight communication trees by considering the link weight shown in (2-11). The operational duration of each cover tree is set with a fixed value $\tau$, unless there is a member of a cover tree that has energy enough to carry its duties for duration $\tau_i$, where $\tau_i < \tau$. In this case, the operational duration of this cover tree is set as equal as the life of this sensor, i.e. $\min(\tau, \tau_i)$.

**Maximum Connected Load-balancing Cover Tree (MCLCT):** To control the connectivity and coverage of sensors on the discrete points of interest (DPOIs), the authors in [91] proposed the MCLCT, which comprises two components: Coverage-Optimising Recursive heuristic (COR) and Probabilistic Load-Balancing strategy (PLB). The aim of the COR is to find the maximum number of disjoint sets of sensors that can fully cover all of the DPOIs. On the other hand, the PLB aims to balance the sensor’s load through a transmission probability.

Based on the assumption, the positions of the sensors and DPOIs are known. The coverage over the DPOIs, therefore, can be measured by considering the distance between a sensor and a point. The covered DPOIs are stored into a set of an individual coverage of each sensor. Considering the set, the COR contributes a disjoint set by recursively collecting sensors until all of the DPOIs are fully covered. Each disjoint set is then assigned a finite operational period. Furthermore, as well contributing the disjoint set in the beginning state, the COR is also able to recover incomplete coverage due to malfunctioning of active sensors by replacing them with other residual sensors that have been not included in any set yet. For the PLB,
considering a forwarding probability, a sensing sensor selects a parent candidate to which its data is transmitted. The forwarding probability is a proportion of the energy-to-distance ratio.

### 2.6.2.2 Discussion

Based on a greedy heuristic algorithm, the CWGC approach can rapidly find a set of connected-cover trees. Sensors with a high potential to cover many DPOIs are greedily selected as sensing nodes; meanwhile the connected-cover trees with the shortest path are greedily constructed. Furthermore, as adopted for the distributed manner, the CWGC has low overhead because local information e.g., residual energy is mainly used to reflect the circumstances of a network. The quality of connected coverage is excellent because the algorithm has a mechanism that can qualify both connectivity and coverage. Moreover, the energy balancing is also excellent as the link weights are determined from the sensor residual energy, which directly reflects the status of the sensors. Hence, only currently potential sensors will be selected.

The MCLCT allows active sensors to balance the traffic load by a forwarding probability. The forwarding probability is composed of a sensor’s residual energy and the distance between the transmitter and receiver. Obviously, the sensed data is to be forwarded to potential nodes all of the time. Therefore, energy consumption among the sensors can be equally balanced. Furthermore, the approach can recover incomplete coverage due to abnormal functioning of an active sensor. Some sensors in sleep mode will be activated to replace abnormal ones in order to maintain the quality of connected coverage. This gives the MCLCT fault tolerance.

However, the CWGC works under many constraints e.g. the sensor locations are required as well as the time synchronisation among the sensors. The MCLCT is one of algorithms designed based on the centralised manner. According to the greedy algorithm of both approaches (CWGC and MCLCT), the best solution to find subsets of sensors to cover targets is guaranteed. Moreover, they are sensitive to position errors that may be incurred by the sensors.
Table 2-3: Achievements and Drawbacks of the Heuristic based Approaches

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Achievement(s)</th>
<th>Drawback(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWGC [18]</td>
<td>- Fast in finding cover-tree sets by greedy heuristic algorithm</td>
<td>- Not guaranteed to obtain the maximum number of cover-tree sets</td>
</tr>
<tr>
<td></td>
<td>- Low overhead as the link weight originally determined from sensors themselves</td>
<td></td>
</tr>
</tbody>
</table>
| MCLCT [91] | - Balancing the traffic load among the disjoint sets while connectivity and coverage are maintained | - Not guaranteed to obtain the maximum number of disjoint sets
|            | - Connected coverage recovery                                                  | - Suitable only for centralised networks                                   |

2.6.3 Stochastic based strategy

2.6.3.1 Ant colony optimisation approaches

The ant colony optimisation (ACO) algorithm imitates the natural behaviour of real ants, which find a route between their nest and a food source. At the beginning, each ant randomly selects a path and lays a pheromone trail whilst it is walking. The ants also put down the pheromone trail on their way back to their nest after finding food. As the pheromones evaporate over time, each ant starts travelling again by choosing the path with the most intense amount of pheromones and it reinforces the path’s pheromones by putting down its own. The travelling is repeatedly performed until the ant colony can find the shortest path from their nest to the food source [95]. For example, the ACO was applied to the Traveling Salesman Problem (TSP) [96].

**Ant-Colony-Based Scheduling Algorithm (ACB-SA):** The authors in [97] adopted the ACO algorithm for sensor scheduling and improved the algorithm’s performance by applying a new initialisation method and modifying the construction graph. In the initialisation stage of each time slot, the ACB-SA collects the number of DPOIs covered by each sensor and calculates the pheromone trail which is given by:

$$\gamma_i(t_s) = e_i \cdot |P_i|,$$

(2-12)

where $\gamma_i(t_s)$ is the amount of pheromone trail at time $t_s$, $e_i$ is the residual energy and $|P_i|$ is the number of the DPOIs that are covered by sensor $s_i$. The initial amount of the pheromone trail allows ants to efficiently select the potential sensors. In the construct graph, unlike the
conventional ACO algorithm, each ant selects and adds sensors one at a time into the covering set until all of the DPOIs are covered. The probability used in the sensor selection is given by:

\[ p_i = \frac{\gamma_i}{\sum_j \gamma_j} \]  

(2-13)

where \( \sum_j \gamma_j \) is the total value of the pheromone trail of the sensors.

**ACO for Maximizing the Number of Connected Covers (ACO-MNCC):** The aim of the work in [27] was mainly to find the maximum number of disjoint sets of connected covers. The ACO-MNCC constructs a graph that represents the assignments of the devices, including sensors and sinks. Considering the high level of probability which consists of heuristic and pheromone information, the ants travel through the constructed graph, resulting in the disjoint sets of the connected covers. The probability of an unassigned device \( j \) to a subset of the assignments is calculated by:

\[ p_i(j) = \frac{\gamma_i(j)[\delta_i(j)]^\beta}{\sum_n \gamma_n(j)[\delta_n]^\beta} \]  

(2-14)

where \( \beta > 0 \) is a predefined parameter, \( \delta_i(j) \) is the heuristic value, and \( \gamma_i(j) \) is the pheromone value. The heuristic information reflects the quality of coverage and connectivity of each subset. The heuristic value is given by:

\[ \delta_i(j) = \begin{cases} \Delta k_i, & \text{if } j \text{ is a sensor}, \\ \Delta c_i, & \text{if } j \text{ is a sink}, \end{cases} \]  

(2-15)

where \( \Delta k_i \) and \( \Delta c_i \) are increments of the coverage percentage and the proportion of the collected sensors after being involved in a coverage set. On the other hand, the pheromone trail, which reflects the potentiality of each path that the ants walk through, is given by:

\[ \gamma_i(j) = \begin{cases} \frac{1}{|\Gamma_i|} \sum_{l \in \Gamma_i} \gamma(j, l), & \text{if } \Gamma_i \neq \emptyset \text{ and } i = 1, 2, 3, ..., N_{as} \\ \gamma_0, & \text{otherwise}, \end{cases} \]  

(2-16)
where $\Gamma_i$ a set of assignments, $\gamma_0$ is the initial pheromone value and $\gamma(j, l)$ is the pheromone between two devices, $j$ and $l$, which can be updated from both local and global levels.

**Three Pheromones ACO (TPACO):** The authors in [98] proposed a novel ACO algorithm using three types of pheromones: two global pheromones and one local pheromone. The first global pheromone, $\tau_{AS}$, represents the potential sensors that are able to cover DPOIs. This pheromone is also utilised to balance the sensors’ energy by considering the residual energy. Another global pheromone, $\tau_{NoAS}$, optimises the required number of covering sensors per DPOI. This type of global pheromone allows an ant to select fewer covering sensors, as this is more likely to conserve energy. Both global pheromones are updated when every ant in a colony has ended its journey. On the other hand, the local pheromone, $\tau_{SS}$, reflects the cumulative number of sensors selected by an ant. This pheromone is updated every time an ant selects a sensor; it is then used in consideration by the following ant.

### 2.6.3.2 Discussion

In the ACO-MNCC, the excessive computation needed to construct an assignment graph is avoided by adding one more subset of the assignments to the best-so-far solution ($N_{as} = N'_{as} + 1$), where $N_{as}$ and $N'_{as}$ are the number of assignments and the best-so-far solution respectively. As the ACO-MNCC supports various kinds of sensors, the relationship between the communication and sensing coverage ranges is flexible because the approach verifies these properties separately. The quality of connected coverage is excellent as the approach favours the potential path, which indicates a high quality of connected coverage through the pheromone trail. Ants, therefore, are able to seek the path that has the most intense value. Moreover, the approach minimises sensor redundancy from disjoint sets of sensors, so the energy consumption can be reduced.

However, balancing the energy consumption among the sensors in the network is not necessarily considered. This can result in the energy of some sensors depleting quicker than others. The time complexity is given by $O(N_{co} \cdot N_{ant} \cdot N_{as} \cdot (N - 1))$, where $N_{co}$ is the number of colonies, $N_{ant}$ is the number of ants, and $N$ is the number of sensors, which is higher than other approaches in the same category. Therefore, the ACO-MNCC cannot be scaled.

Compared to the conventional ACO algorithm, the ACB-SA improved the time convergence for the ants to find a disjoint set of active sensors covering all of the DPOIs by the novel
initialisation and graph construction methods. The quality of the coverage produced by the set is excellent as the approach minimises the number of active sensors. It also yields efficient energy use. Furthermore, the energy consumption is well balanced among the sensors by encouraging sensors with high residual energy to be active.

TPACO is similar to ACB-SA, except that it uses three pheromones for the ants to find the best solution. Using these pheromones, the TPACO can minimise the number of active sensors covering all of the DPOIs. Moreover, the sensors consume energy equally because those with high residual energy are favoured by the ants. As mentioned above, the TPACO considers covering all of the DPOIs with the minimum number of active sensors. Also the sensors are well rotated by boosting one of the global pheromones with their residual energy. Therefore, it yields excellent energy efficiency and balance. Furthermore, the TPACO has an algorithm to reduce redundant coverage, so the coverage efficiency is excellent.

However, neither the ACB-SA nor the TPACO guarantee the connectivity between active sensors and their neighbours, and between active sensors and sinks. In particular, in the DPCC problem, it is crucially important to consider the connectivity, as the DPOIs are usually separated from each other and this necessarily causes the active sensors to be disconnected. These approaches are also classified as moderate in terms of the local error tolerance due to the locations of the sensors and DPOIs being assigned without any verification. In particular, the coverage for the DPOIs requires precise location information. All of the approaches stated here are performed in a centralised manner. This is in contrast to a principal nature of WSNs that requires being self-organised and highly scalable.

Table 2-4: Achievements and Drawbacks of the Stochastic based Approaches

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Achievement(s)</th>
<th>Drawback(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACB-SA [97]</td>
<td>- The new initialisation method for the pheromone trail</td>
<td>- No connectivity guarantee</td>
</tr>
<tr>
<td></td>
<td>- The new construction graph reflecting the energy-efficient coverage problem</td>
<td>- Suitable only for centralised networks</td>
</tr>
<tr>
<td>ACO-MNCC [27]</td>
<td>- Avoiding excessive computation to obtain the maximum number of disjoint sets</td>
<td>- Suitable only for centralised networks</td>
</tr>
<tr>
<td></td>
<td>by adding one more connected cover from the best-so-far solution</td>
<td>- High computational cost</td>
</tr>
<tr>
<td></td>
<td>- Reducing redundancy by the local search procedure</td>
<td></td>
</tr>
<tr>
<td>TPACO [98]</td>
<td>- Three pheromones resulting in energy balancing, minimising number of active</td>
<td>- No connectivity guarantee</td>
</tr>
<tr>
<td></td>
<td>sensors</td>
<td>- Suitable only for centralised networks</td>
</tr>
</tbody>
</table>
2.6.4 Partition based strategy

2.6.4.1 Partition based approaches

Centralized Randomized $k$-Coverage (CERACC$k$): CERACC$k$ maintains the quality of connected coverage using a virtual partition; it was suggested by the authors of [77]. It uses Lens as a target area in which sensors are selected. The Lens stands for the lens area which is an intersection area between two adjacent sensing discs (see Figure 2-16(a)). In fact, lenses can be a particular area between common edges of adjacent equilateral triangles in a hexagonal cell (see Figure 2-16(b)). In this method, only $k$ sensors within the lens area will be selected. An active sensor in the lens is able to cover both sides of the adjacent equilateral triangles if the triangles’ sides are equal to the sensing radius $R_s$. Since a hexagon consists of 6 pieces of equilateral triangles, it can be divided into two sets of the 3-lenses. The sets are $\{L_1, L_3, L_5\}$ and $\{L_2, L_4, L_6\}$ in which sensors are selected to cover a whole hexagonal cell.

In the algorithm of the CERACC$k$ consists of two phases: field partitioning and sensor selecting. In the former, the monitored field is partitioned into a number of lenses. The sensors are grouped into lens areas by the partitioning. Then, $k$-sensors are selected to be active from each lens in the sensor selecting phase. A set of active sensors can fully $k$-cover the entire monitored area if the sensor selection is successful for all of the lenses. These processes are performed repeatedly in each operational round.

Figure 2-16: Lens area used in the Lens method (a) Intersection area called Lens (b) Two sets of Lens to cover the whole hexagonal cell
A Coverage-Aware Clustering Protocol (CACP): CACP [99] combines two efficient techniques – network clustering and sensor scheduling – to minimise energy consumption. The cluster heads (CHs) are selected by a timer mechanism and they then divide their own cluster area into small virtual pieces in the shape of a regular triangle tessellation (see Figure 2-17). The length of each side of the regular triangle is equal to $\sqrt{3}R_s$, where $R_s$ is the sensing radius, and each vertex is defined as a target point $n$ and target area $a(n)$, which is rotated in

![Figure 2-17: The triangle tessellation layered around a cluster head [100]](image)

![Figure 2-18: Sponsors and covering area on a target area [100]](image)

$n$ : a target point
$S_i$ : a candidate sponsor
every round. Every target area is covered by certain sensors called sponsor nodes $s_i$. For example (see Figure 2-18), the target area, which is centred at point $n$ with radius $R_s$, can be fully covered by two sets of sponsors as $S_1 = \{s_1, s_2, s_3, s_4\}$ or $S_2 = \{s_1, s_2, s_4\}$. The CACP algorithm precisely selects $S_2$ to be active, which is the best solution for this target area. Obviously, $s_3$ then enters into sleep mode. In addition, the sensor activation process is carried out as a layer manner which starts at the innermost target points to the outermost ones.

2.6.4.2 Discussion

The CERACC$k$ algorithm is the centralised manner, which the sensors are managed by a base station (BS). By the BS, a grid is drawn on the monitored field as a virtual partition in order to make lens areas in which sensors have a chance of being selected. This algorithm can maintain excellent quality of the $k$-coverage and connectivity because of the proper structure of the partition. This structure can also limit the number of active sensors, so that the energy consumption is minimised. In addition, as sensors are grouped into lenses, overheads in terms of time complexity and massage complexity are low because only local information needs to be investigated.

However, The CERACC$k$ algorithm selects sensors only in the lens areas, which is a cause of an unequal chance of the sensors being selected. This is because the lenses’ areas are a fraction of the total monitored area. It means that the rest of sensors, which are outside of the lenses’ areas, having no chance to be selected. Although the algorithm distributes the chance by randomly assigning the original position of the partition and re-partitioning the monitored field, an equal chance of sensors being selected is not guaranteed.

Contrary to The CERACC$k$ algorithm, the CACP algorithm is distributed manner. Its hierarchical network allows sensors to self-organise. Energy consumption can be minimised by clustering and sensor scheduling. However, the CACP algorithm is location error sensitive due to using points as targets. Since the algorithm grants a CH to assign the target points based on its position within a cluster, the target points’ locations may be wrong if the CH’ position is inaccurate. Furthermore, quality of the connected coverage may be not guaranteed. In the algorithm, a sensor decides itself to be a CH if it not receives any announcement from another CH in neighbourhood within a certain period. By this way, this CH could thus be isolated from the rest of the network. This is against the quality of the connected coverage.
Table 2-5: Achievements and Drawbacks of the Partition Based Approaches

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Achievement(s)</th>
<th>Drawback(s)</th>
</tr>
</thead>
</table>
| CERACCₖ [77] | - Complete connected k-coverage using lenses  
- Minimum number of active sensors  
- Low time complexity | - Unequal chance of sensors being selected  
- Need to repartition every operational round |
| CACP [99] | - Distributed system  
- Hierarchical network  
- Minimum energy consumption | - Sensitive to location error  
- Quality of connected coverage may be not guaranteed |

2.7 Summary

The relevant background regarding the sensing coverage and the network connectivity has been explained in this chapter. Moreover, a review of sensor scheduling strategies in literature has been given in order to reflect the most current techniques in the field and to identify gaps. In the controlling quality of connected coverage, there are three different problems according to the types of targets, namely discrete point connected coverage (DPCC), partial area connected coverage (PACC), and entire area connected coverage (EACC). Each of the problems has a particular scenario that must be taken into account in order to control the sensing coverage and network connectivity. The relevant properties, including the sensor devices, their sensing ability and connectivity and the relationship between the sensing and communication of a sensor, have been described in detail. Based on these properties, we can verify whether the quality of both coverage and connectivity meets the requirements.

Furthermore, the state-of-the-art existing sensor scheduling approaches have been reviewed and briefly expressed in this chapter. The approaches have been categorised into different strategies and their achievements and drawbacks have been compared. There are four major issues that have not been yet fully addressed in literature; e.g., an unequal chance of sensors being active, a high degree of coverage redundancy over requirement, energy imbalance among the sensors, and high computational time complexity. These issues can significantly degrade the network’s performance. Therefore, this thesis approaches and tries to deal with these issues.
### Table 2-6: Comparison of the Approaches in the Connected Coverage Assurance for Sensor Scheduling

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Complexities</th>
<th>Constraints</th>
<th>Performances</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Message</td>
<td>LC</td>
</tr>
<tr>
<td>CCP [78]</td>
<td>$O(N^2)$</td>
<td>$O(N + N_{sleep})$</td>
<td>Required</td>
</tr>
<tr>
<td>OGDC [71]</td>
<td>$O(N^2)$</td>
<td>$O(N_a)$</td>
<td>Required</td>
</tr>
<tr>
<td>DBCCR [88]</td>
<td>$O(1)$</td>
<td>$O(N + N_a)$</td>
<td>Free</td>
</tr>
<tr>
<td>ROC [93]</td>
<td>$O(N^2)$</td>
<td>$O\left(\frac{T}{t_p}(N + N_a) + \frac{T}{t_p}N_i\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>CWGC [18]</td>
<td>$O(N \cdot N_{ant}^2)$</td>
<td>$O\left(\frac{N + N_s}{2} + N_r\left(\frac{N_D + 1}{2}\right)\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>MCLCT [91]</td>
<td>$O(N \cdot \log(N))$</td>
<td>$O\left(N \left(\frac{N_D + 1}{2}\right)\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>ACB-SA [97]</td>
<td>$O(N_{co} \cdot N_{ant} \cdot N)$</td>
<td>$O\left(N \left(\frac{N_D + 1}{2}\right) \cdot t_s\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>ACO-MNCC [27]</td>
<td>$O\left(N_{co} \cdot N_{ant} \cdot N_{as} \cdot (N - 1)\right)$</td>
<td>$O\left(N \left(\frac{N_D + 1}{2}\right)\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>TPACO [98]</td>
<td>$O(N_{co} \cdot N_{ant} \cdot N)$</td>
<td>$O\left(N \left(\frac{N_D + 1}{2}\right) \cdot t_s\right)$</td>
<td>Required</td>
</tr>
<tr>
<td>CERACC [98]</td>
<td>$O(1)$</td>
<td>$O(N + N_a)$</td>
<td>Required</td>
</tr>
<tr>
<td>CACP [99]</td>
<td>$O\left(\max(N_i + N_{as}^2)\right)$</td>
<td>$O(N + N_a)$</td>
<td>Required</td>
</tr>
</tbody>
</table>

List of symbols: $N$ is the number of all deployed sensors in the network; $N_a$ is the number of active sensors; $N_{ant}$ is the number of ants; $N_{as}$ is the number of assignments; $N_D$ is the number of hops of the network dimension; $N_i$ is the number of neighbours of sensor $s_i$; $N_{as}$ is the number of iterations; $N_{DPOI}$ is the number of DPOIs; $N_r$ is the number of sensors performing relaying task; $N_s$ is the number of sensors performing sensing task; $T$ is the network lifetime; $t_s$ is the time slot; $t_p$ is the period of probing.

List of abbreviations: LC is location; SC is sensor capacities; CSR is communication-sensing relationship; SYN is synchronisation; QCV is quality of coverage; QCN is quality of connectivity; EC is energy conservation; EB is energy balance; SFA is sensor failure avoidance; LET is location error tolerance.
### Table 2-7: Properties of the Approaches in the Connected Coverage Assurance for Sensor Scheduling

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCP [78]</td>
<td>EACC</td>
<td>Geo</td>
<td>Event</td>
<td>Distributed</td>
<td>Hete No Random</td>
<td>Intersection</td>
<td>∃k Implying</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OGDC [71]</td>
<td>EACC</td>
<td>Geo</td>
<td>Round</td>
<td>Distributed</td>
<td>Hete No Random</td>
<td>Intersection</td>
<td>Broadcasting</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBCCR [88]</td>
<td>PACC</td>
<td>Geo</td>
<td>Round</td>
<td>Distributed</td>
<td>Homo No Random</td>
<td>Intersection</td>
<td>≥1 Additional</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>ROC [93]</td>
<td>EACC</td>
<td>Geo</td>
<td>Round</td>
<td>Distributed</td>
<td>Homo No Random</td>
<td>Intersection</td>
<td>≥1 Implying</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Probabilistic</td>
<td>Binary</td>
</tr>
<tr>
<td>CWGC [18]</td>
<td>DPCC</td>
<td>Heu</td>
<td>Round</td>
<td>Distributed</td>
<td>Homo No Random</td>
<td>Distance</td>
<td>≥1 Additional</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>MCLCT [91]</td>
<td>DPCC</td>
<td>Heu</td>
<td>Round</td>
<td>Centralised</td>
<td>Homo No Random</td>
<td>Distance</td>
<td>≥1 Additional</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>ACB-SA [97]</td>
<td>DPCC</td>
<td>Stoc</td>
<td>Round</td>
<td>Centralised</td>
<td>Hete No Random</td>
<td>Distance</td>
<td>N/A BINARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Probabilistic</td>
<td>≥1</td>
</tr>
<tr>
<td>ACO-MNCC [27]</td>
<td>EACC</td>
<td>Stoc</td>
<td>Round</td>
<td>Centralised</td>
<td>Hete No Random</td>
<td>Proportion</td>
<td>≥1 Independent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>TPACO [98]</td>
<td>DPCC</td>
<td>Stoc</td>
<td>Round</td>
<td>Centralised</td>
<td>Hete No Random</td>
<td>Distance</td>
<td>N/A BINARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Probabilistic</td>
<td>≥1</td>
</tr>
<tr>
<td>CERACC [77]</td>
<td>EACC</td>
<td>Part</td>
<td>Round</td>
<td>Centralised</td>
<td>Homo No Random</td>
<td>Intersection</td>
<td>≥k Implying</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
<tr>
<td>CACP [99]</td>
<td>EACC</td>
<td>Part</td>
<td>Round</td>
<td>Distributed</td>
<td>Homo No Random</td>
<td>Intersection</td>
<td>≥1 Implying</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Circular Binary</td>
<td>Binary</td>
</tr>
</tbody>
</table>

Note that Cov. Prob. is coverage problems; Cov. Tech. is Coverage controlling technique; Alg. Manner is algorithm manner; Capa. is sensor capacities; Mob. is mobility; Depl. sensor deployment.
Chapter 3

A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

In this chapter, the first sensor scheduling method that achieves complete connected coverage is proposed. This method uses a hexagon tessellation as a virtual partition in a monitored field. The virtual hexagon partition divides the sensors into groups called hexagonal cells and maintains the quality of connected coverage produced by the sensors between the cells. The detail is described below.

3.1 Introduction

In Section 2.1 the problems in regard to connected coverage assurance are stated. This research, including this chapter and the following chapters, focuses on the Entire area connected coverage (EACC) problem. The EACC problem is considered to be most challenging because of its problem space. In addition, a solution to this problem can be applied to other relevant problems. Based on the problem statement regarding the EACC problem expressed in Section 2.1.3, the principal problem in this research is defined as:

<table>
<thead>
<tr>
<th>The principal problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Given a monitored area $\mathcal{A}$, and a set $S$ of the deployed sensors $(S = {s_i</td>
</tr>
</tbody>
</table>
Chapter 3. A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

3.1.1 Objectives

The objectives of this chapter are as follows:

- To design a sensor scheduling method that overcomes the purpose of the principal problem
- To analyse the performance of the proposed method and validate the results using a simulation
- To compare the proposed method with other existing algorithms

3.1.2 Contributions

A sensor scheduling method called the 6-Triangle (6-Tri) method is introduced and compared to other existing algorithms – Centre and Lens – proposed by authors in [100] and [77], respectively. The 6-Tri method uses a hexagon tessellation as a virtual partition in order to divide the sensors into groups of hexagonal cells. A hexagonal cell is composed of 6 equilateral triangles in which the sensors are further divided into sub-groups. An equilateral triangle is exploited as a target area from which $k$ sensors will be selected to be active, whilst the other unselected sensors simply switch to sleep mode in order to conserve their energy.

The novelty of this sensor scheduling method is that using the virtual hexagon partition can achieve connected coverage assurance at the required level. In addition, as the 6-Tri method is designed in a decentralised manner, the sensors organise themselves into the virtual hexagon partition and then select themselves to be active. This means that the method has very low overheads in terms of time complexity and message complexity. Moreover, energy consumption is equally balanced among the sensors, as there is an equal chance of being selected for all of the sensors.

3.1.3 Chapter organisation

The remaining sections of this chapter are organised as follows: In Section 3.2, a network model together with definitions of the sensing model and communication model are stated. Section 3.3 illustrates the structure of a virtual hexagon partition and describes how the sensors determine their position in the partition. Then, Section 3.4 explains how the 6-Tri
method selects sensors from each cell. The performance of the method is then evaluated and discussed in Section 3.5. Finally, Section 3.6 concludes this study.

### 3.2 Network Model

In the network model, a monitored area is a square area with dimensions equal to $\mathcal{A} = W \times L$. The base station (BS) is at the centre of this area, BS($x = 0, y = 0$). Sensors are randomly and uniformly deployed with network density $\rho$. Sensors are homogeneous and are assumed that they know their own coordinates $s(x_i, y_i)$ originated at BS. They also have sensing and communication properties as:

**Definition 3-1 (Sensing disc and coverage property).** Let $s_i$ be a sensor $i$, $(S = \{s_i|i = 1, 2, ..., N\})$, located at coordinates $(x_i, y_i)$ in the monitored area $\mathcal{A}$. Let $R_s$ be a sensing radius which every sensor has the same size. The sensing disc is the coverage area of $s_i$, which is denoted by $\mathcal{D}(s_i, R_s)$ or $\mathcal{D}_i$ centred at sensor’s position and its area denoted by $\mathcal{A}_\mathcal{D}$. The coverage property is that any point $q$ is covered by $s_i$ if $q \in \mathcal{D}_i$. In addition, $q$ is $k$-covered if it is in an intersection coverage area of $k$ sensors, $q \in \bigcap_{i=1}^{k} \mathcal{D}_i$.

**Definition 3-2 (Connectivity property).** Let $R_c$ be the communication radius. The connectivity property of any pair of the sensors is that sensors $s_i$ and $s_j$ are connected to each other if the Euclidean distance between them is at most the communication radius, i.e. $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \leq R_c$. The relationship between the communication and sensing radii is $R_c \geq 2R_s$.

### 3.3 A Virtual Hexagon Partition and Position Determination

A virtual partition is used to produce particular areas as targets for sensor selection across the monitored area. The partition consists of hexagon cells that divide deployed nodes into groups, based on their geographical position. Since these hexagonal cells have their size related to a sensing radius, the distances of sensors in adjacent cells are limited by this radius. This section explains a virtual hexagon partition and how to produce target areas on a monitored area as well as how to recognise positions upon this virtual partition by sensors themselves.
Chapter 3. A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

3.3.1 Fundamental hexagonal cell

A virtual hexagon cell has a set of six pieces of equilateral triangles with an angle $\theta_0 = \frac{\pi}{3}$. The hexagonal cell is completely covered by the sensing disc, when the centre of the disc is on the centre of the hexagonal cell and its radius is equal to the sensing radius (see Figure 3-1).

**Definition 3-3 (Hexagonal Cell).** The hexagonal cell is the fundamental unit of the virtual partition in hexagon shape. It is composed of six pieces of the equilateral triangle, $P_z$ for $z = 0, 1, ..., 5$. The hexagonal cell is denoted by $H(O_{uv}, P_z, R_s)$, where $O_{uv}$ is the cell’s centre located at alignments $u$ and $v$ in the virtual partition.

**Definition 3-4 (Virtual Hexagon Partition).** The virtual hexagon partition is as a hexagon tessellation virtually covering over the monitored area. It is composed of the consecutive hexagonal cells, where the original cell is at the base station.

The distance from cell’s centre to any vertex is the hexagon radius which is equal to the sensing radius($R_s$). The cell’s centre is located at coordinate $(x_O, y_O)$ in the monitored field. The relationship between the coordinates in the monitored field and alignment in the virtual hexagon partition is given by:

\[
\begin{align*}
    x_O &= 3uR_s \cos \theta_0, \\
    y_O &= \begin{cases} 
    2vR_s \sin \theta_0, & \text{if } u \text{ is even}, \\
    (2v + 1)R_s \sin \theta_0, & \text{if } u \text{ is odd}, 
    \end{cases}
\end{align*}
\]

Figure 3-1: A hexagonal cell (a) dimension of the cell and six pieces of equilateral triangle (b) different parts in the cell
Chapter 3. A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

where $u$ and $v$ are alignments on $x$ axis and $y$ axis, respectively, for $u = (\pm)\{0, 1, 2 \ldots, U\}$ and $v = (\pm)\{0, 1, 2 \ldots, V\}$, where $(\pm)$ stands for the quadrant of the cell compared to the BS, $U \leq \frac{1}{3R_s \cos \theta_O} \frac{W}{2}$ and $V \leq \frac{L}{R_s \sin \theta_O}$.

3.3.2 Position determination

Position determination is that every sensor identifies its own position on a cell and piece of the virtual hexagon partition. The monitored area can be divided into four symmetric quadrants – $Q_1$ to $Q_4$, when the BS is at the original of the network (it can be anywhere). Figure 3-2 shows the virtual hexagon partition of $Q_1$ representing three other quadrants. The centre of the partition is set by the original cell, $\mathcal{H}(O_0, \mathcal{P}_z, R_s)$, where the BS is placed. The centres of other cells are aligned on $x$ axis at $3R_s \cos \theta_O, 6R_s \cos \theta_O, \ldots, 3uR_s \cos \theta_O$. On the other hand, these cells are aligned on $y$ axis in different ways depending on their relationship with the alignment number on $x$ axis. The relationship of alignments between $x$ and $y$ axes, says column and row (column-row) respectively, is even-even and odd-odd manners. In the former, the centres of the hexagonal cells are aligned on $y$ axis at $0, 2R_s \sin \theta_O, 4R_s \sin \theta_O, \ldots, 2v \sin \theta_O$, when $u$ is even, so that the relationship between $u$ and $v$ is even-even. For the latter, hexagonal cells are aligned at $R_s \sin \theta_O, 3R_s \sin \theta_O, \ldots, (2v + 1)R_s \sin \theta_O$, when $u$ is odd. Thus, the relationship between $u$ and $v$ is odd-odd.
In the initial phase, every sensor calculates its position on the virtual hexagon partition. A sensor firstly identifies the lower bounds of column and row alignments. Based on its coordinate of \( s(x_i, y_i) \), the lower bounds \( \bar{u} \) and \( \bar{v} \) are given by:

\[
\bar{u} = \left\lfloor \frac{|x_i|}{(3R_s \cos \theta_O)} \right\rfloor, \quad \bar{v} = \left\lfloor \frac{|y_i|}{(R_s \sin \theta_O)} \right\rfloor.
\]

(3-2)

The pair of results of Equation (3-2) has four possible cases: (i) even-even, (ii) odd-odd, (iii) even-odd and (iv) odd-even. By these cases, the sensor can find candidates of hexagonal cells surrounding its position. As every centre of the hexagonal cells is either on even-even or odd-odd alignments, the possible candidates – \( O(x_a, y_a) \) and \( O(x_b, y_b) \) – are located at:

\[
(x_a, y_a) = \begin{cases} 
(\pm) \left( \frac{3}{2} \bar{u}R_s, \frac{\sqrt{3}}{2} \bar{v}R_s \right), & \text{(i) or (ii)} \\
(\pm) \left( \frac{3}{2} \bar{u}R_s, \frac{\sqrt{3}}{2} (\bar{v} + 1)R_s \right), & \text{(iii) or (iv)} 
\end{cases}
\]

\[
(x_b, y_b) = \begin{cases} 
(\pm) \left( \frac{3}{2} (\bar{u} + 1)R_s, \frac{\sqrt{3}}{2} (\bar{v} + 1)R_s \right), & \text{(i) or (ii)} \\
(\pm) \left( \frac{3}{2} (\bar{u} + 1)R_s, \frac{\sqrt{3}}{2} \bar{v}R_s \right), & \text{(iii) or (iv)} 
\end{cases}
\]

(3-3)
Figure 3-3 shows four possible cases of two cell candidates that any sensor could have. From these cases, every sensor is surrounded by only two candidates, so that the sensor can decide itself by determining the closet one to which it belongs.

After identifying the closet hexagonal cell, the sensor sets this cell’s centre $O_{uv}$ as its reference point and examines a piece of the equilateral triangle of this cell. Once again, an area around the reference point (centre of the cell) is divided into four quadrants. An angle ($\theta_i$) of the sensor to the cell’s centre is given by:

$$\theta_i = \tan^{-1}\left(\frac{|y_o - y_i|}{x_o - x_i}\right).$$

Hence, the piece ($P_z$) of the equilateral triangles in the hexagonal is given by:

$$P_z = \begin{cases} 
\frac{\theta_i}{\theta_o} & x_i \geq x_o \text{ and } y_i \geq y_o, \\
\frac{\pi - \theta_i}{\theta_o} & x_i < x_o \text{ and } y_i \geq y_o, \\
\frac{\pi + \theta_i}{\theta_o} & x_i < x_o \text{ and } y_i < y_o, \\
\frac{2\pi - \theta_i}{\theta_o} & x_i \geq x_o \text{ and } y_i < y_o.
\end{cases}$$

Now, the sensor realises its own position on the virtual hexagon partition $H(O_{uv}, P_z, R_s)$, including centre’s position of the cell ($O_{uv}$), the piece of the equilateral triangle ($P_z$) and a given radius of the cell ($R_s$).

### 3.4 6-Triangle (6-Tri) Sensor Scheduling Method

In order to conserve network energy, a minimum number of sensors are active, while the connected coverage is maintained. Based on the virtual hexagon partition, the grouped sensors from each target area are selected to be active. This section analyses a sensor
scheduling method called Centre and then describes the proposed selection method called 6-Tri method.

### 3.4.1 Centre method

According to Definition 3-3, obviously, the area of the hexagonal cell is less than the area of the sensing disc. The hexagonal cell is fully covered by only one sensor if it is at the centre of the cell. Thus, the Centre method uses the cell’s centre as the target point to select sensors. Selecting sensors at exact points is incredibly difficult in reality because sensors are randomly deployed. Therefore, the sensors with the closest target points will have more chance of being selected than the others.

**Theorem 3.1** Any active sensor at the centre of a hexagonal cell can cover a whole area of the cell by its sensing disc, \( \mathcal{H}(O_{uv}) \subseteq \mathcal{D}_i \mid d(O_{uv}, s_i) = 0 \). Furthermore, any pair of active sensors can communicate with each other such that \( R_c \geq 2R_s \). Therefore, the complete connected coverage for the whole monitored area is guaranteed if every hexagonal cell is covered.

**Proof.** The area of the equilateral triangle is given by \( \mathcal{A}_{Tri} = \frac{\sqrt{3}}{4} R_s^2 \). On the other hand, a sector area of the sensing disc with an angle at the centre \( \theta_0 \) is given by \( \mathcal{A}_{Sec} = \frac{\pi}{6} R_s^2 \). Obviously, we have \( \mathcal{A}_{Tri} < \mathcal{A}_{Sec} \) and \( \mathcal{A}_{Tri} \subset \mathcal{A}_{Sec} \) for every piece of the equilateral triangles and sectors. Thus, any point in the hexagonal cell is covered by the sensing disc when the centres of both the hexagonal cell and the sensing disc are at the same position. On the other hand, the distance between the centres of adjacent cells is given by \( d(O_{u0v0}, O_{u1v1}) = \sqrt{3}R_s \). Therefore, the active sensors in both cells have a connection because of \( d(O_{u0v0}, O_{u1v1}) < R_c \).

However, the Centre method has a serious problem in guaranteeing the sensing coverage due to the problem of selecting sensors in the right places. According to Theorem 3.1, a sensor must be selected at the centre of a hexagonal cell in order to fully cover the entire area of the cell. It is almost impossible to select sensors at the centre of every cell, especially in such random networks. Therefore, the Centre method cannot guarantee the quality of sensing coverage for the entire network.
3.4.2 6 Triangles (6-Tri) method

Based on a hexagon cell, a piece of equilateral triangle is considered as a target area. Sensors within each piece of the equilateral triangles have to be selected. Any sensor in a triangle can cover any point of this triangle area. Therefore, to $k$-cover the entire hexagonal cell, each target area is required to have $k$ active sensors.

**Theorem 3.2** Any sensor within a piece of an equilateral triangle is able to completely cover this triangle area. Consequently, $k$ sensors within the equilateral triangle are able to $k$-cover the whole triangle area. Therefore, $k$ sensors from each piece of triangles are able to $k$-cover the whole hexagonal cell. Therefore, the connected coverage for the whole monitored area is guaranteed if every hexagonal cell is $k$-covered.

**Proof.** The distance of any pair of points within an equilateral triangle is at most $R_s$. Thus, the entire triangle area can be covered by any sensor which is in this triangle. The worst case is considered in order to prove that covering the entire hexagonal cell requires active sensors for every piece of the equilateral triangle. Let $\mathcal{P}_z$ be a piece of the equilateral triangle without any active sensor, called an empty piece. To cover this piece, active sensors from adjacent triangles ($\mathcal{P}_{z-1}$ and $\mathcal{P}_{z+1}$) are required instead. In the worst case, two given sensors are at the farthest vertices from both pieces $\mathcal{P}_{z-1}$ and $\mathcal{P}_{z+1}$, where the distance between them is equal to diameter of the hexagonal cell. A segment area $\mathcal{A}_{seg}$ from a sensing disc of the sensor that can intersect with the empty piece $\mathcal{P}_z$ is given by $\mathcal{A}_{seg} = (\mathcal{A}_{sec} - \mathcal{A}_{tri}) = \left(\frac{\pi}{6} - \frac{\sqrt{3}}{4}\right)R_s^2$, where $\mathcal{A}_{sec}$ and $\mathcal{A}_{tri}$ are the sector area and the triangle area respectively. Thus, the coverage area over the empty piece by the segment areas from both sensors is given by $2\mathcal{A}_{seg} = 2\left(\frac{\pi}{6} - \frac{\sqrt{3}}{4}\right)R_s^2$. Obviously, the covered area on the empty triangle is always less than the triangle area, $\mathcal{P}_z$, $(2\mathcal{A}_{seg} < \mathcal{A}_{tri})$. This implies that covering for the whole hexagonal cell requires $k$ active sensors from every piece of the hexagonal cell. Furthermore, connectivity among these active sensors can be connected because the distance of any pair is less than $R_c$. Therefore, the connected coverage of the whole monitored area can be guaranteed if every hexagonal cell is completely $k$-covered.  \[\square\]
3.4.3 Sensor scheduling

For every scheduling round, based on the 6-Tri method, a set of sensors are recursively selected to be active and others are pushed into sleep mode. As the sensor scheduling is proposed in a decentralisation scenario, a timer mechanism for the sensor selection is suitable. The outstanding advantage of using the timer is that local information exchanged among sensors is not required, which means no extra overheads added into the network operation. Moreover, as the timer has residual energy as a function, where the sensors with high residual energy have more chance of being active than others, the energy consumption can be equally distributed throughout the network’s lifetime. The timer is given by:

$$t_i = t_0\left(\frac{E - \tau e_i}{E}\right)$$

(3-6)

where $t_i$ is timer of sensor $s_i$; $t_0$ is a constant value of time (e.g. $10^{-6}$ sec.); $\tau$ is a random value between 0.9 and 1 to differentiate sensors which have the same level of the residual energy; $e_i$ and $E$ are the residual energy of sensor $s_i$ and the maximum energy at beginning respectively.

The sensors determine the timer for selection competition among themselves, where a set of $k$ sensors will be active from each target area to meet $k$-coverage. They wait for the timer reach to zero. Without hearing any announcement from others, the sensors decide to be active and announce to others within the same group. Otherwise, they terminate the timer if their group has already $k$ sensors. Eventually, $k$ sensors with the highest residual energy from each group are selected themselves to be active.

3.4.4 Sensor selection algorithm

Our algorithm performs simultaneously for each sensor. In the initial phase, sensors primarily realise their position on the virtual hexagon partition themselves before entering a sensor selection phase as shows in Algorithm lines 1 to 5. This phase is performed only once. After finishing the initial phase, every sensor records its own position $\mathcal{H}(O_{uv}, P_z, R_s)$, including the cell’s centre and the piece of the equilateral triangles. Therefore, they are grouped into target areas (equilateral triangles) and are ready for the sensor selection. The sensor selection
process begins at line 7 until line 26. Duty-cycling the sensor is done by a certain interval $T$. At the beginning of every round of the duty-cycle, based on their own residual energy, sensors determine a timer for competition. The sensors in the same cell and same group wait until either the timer reach to zero or receive an announcement of being active from neighbours. The $k$-sensors with the highest residual energy would be active (lines 10 - 14), meanwhile others switch to sleep mode (lines 15 - 25).

**Algorithm: The $6$-Tri method**

| Input: Coordinate $s(x_i, y_i)$ for $i = 1, 2, ..., N$ and BS$(0,0)$, $\theta_0$, $R_s$, $k$ and $E$ |
| Output: A set of active sensors |
| **Initial Phase:** |
| 1 BS broadcasts its coordinates BS$(0,0)$ |
| 2 $s_i$ determines position $\mathcal{H}(O_{uv}, P_z, R_s)$ by using (3-2) to (3-5) |
| 3 $s_i \leftarrow \mathcal{H}(O_{uv}, P_z, R_s)$ |
| 4 $e_i \leftarrow E$ |
| 5 $s_i$ familiarises its neighbours |

**Begin:**

6 Initial Phase ()

7 For ($e_i \geq$ given level) /* e.g. 10% or 0 */

8 $s_i$ determines $t_i$ by (3-6)

9 Waiting ($t_i$) /* waiting timer until zero */

10 If ($t_i == 0$) && ($K < k$)

11 activeSensor($\mathcal{H}(O_{uv}, P_z, R_s)$) $\leftarrow \{s_i | max(e_i)\}$

12 sendAnnouncement ($s_i, \mathcal{H}(O_{uv}, P_z, R_s)$)

13 $k$-sensors($\mathcal{H}(O_{uv}, P_z, R_s)$) $\leftarrow K++$

14 activeOperating($T$) /* this sensor operates in interval $T$ */

15 **Elseif** (recAnnouncement () $== 1$)

16 If ($s_i(\mathcal{H}(O_{uv}, P_z, R_s)) = s_j(\mathcal{H}(O_{uv}, P_z, R_s))$, where $s_i$ is the receiver and $s_j$ is sender)

17 **Else**

18 **End**

19 GettingSleep ($s_i$)

20 End

21 ContinueWaiting ($t_i$)

22 End

23 End

24 $k$-sensors($\mathcal{H}(O_{uv}, P_z, R_s)$) $\leftarrow K++$

25 End

26 End
3.5 Analyses and Simulations

In this section, metrics which are used to evaluate the performance of sensor scheduling methods are discussed. In addition, results from simulation are compared with those from analyses as shows below.

3.5.1 Analyses

- **Chance of being selected:** Generally, sensors deployed in a network are randomly selected to become active to perform their responsibilities for a certain period. In static networks, only the sensors in particular areas, called target areas, can be selected by a scheduling method. Thus, the chance of the sensors being selected is represented by the proportion of target areas over the total area of the monitored field. This metric significantly indicates the performance of a scheduling method of how to balance energy consumption among the sensors. It is given by:

\[
C_h = \frac{\rho \cdot \sum A_{\text{tar}}}{N} \times 100 = \frac{\sum A_{\text{tar}}}{A} \times 100,
\]

(3-7)

where \( \rho \) is the network density; \( N \) is the total number of the sensors; \( A \) is the total area of the monitored field; and \( A_{\text{tar}} \) is a target area, which is given by:

\[
A_{\text{tar}} = \begin{cases} 
\frac{3\sqrt{3}}{2} R_s^2, & A_{\text{hex}} \text{ is hexagon area for Centre,} \\
\left(\frac{\pi}{3} - \frac{\sqrt{3}}{2}\right) R_s^2, & A_{\text{lens}} \text{ is lens area for Lens,} \\
\frac{\sqrt{3}}{4} R_s^2, & A_{\text{tri}} \text{ is triangle area for } 6-\text{Tri.}
\end{cases}
\]

(3-8)

- **Success of sensor selection:** As mentioned above, only a given number of sensors within the target areas can be selected to become active. The success of the sensor selection is the probability that the given number of sensors is successfully selected from each target area. This is the important factor that greatly affects the quality of the connected coverage. Indeed, the success of the sensor selection depends on both the size of the target area and the
number of sensors in each target area. The different sizes of the target areas have been mentioned in (3-8). Since the sensors are deployed randomly across the network, the probability that they are dropped in target areas is Poisson distribution ($P_o$). Furthermore, $k$-sensors are selected within a target area, so that the selection is successful when there are at least $k$ sensors in this area. Let $X$ be the random variable of the number of sensors per target area. The mean number of the sensors in a target area is given by $\rho \cdot \mathcal{A}_{\text{tar}}$, where $\rho$ is network density and $\mathcal{A}_{\text{tar}}$ is a target area. Thus, we are interested in $P(X \geq k, \rho \cdot \mathcal{A}_{\text{tar}})$.

Therefore, the success of sensor selection is given by:

$$\eta = 1 - \left[ P(k - 1; \rho \cdot \mathcal{A}_{\text{tar}}) + P(k - 2; \rho \cdot \mathcal{A}_{\text{tar}}) + \cdots + P(1; \rho \cdot \mathcal{A}_{\text{tar}}) + P(0; \rho \cdot \mathcal{A}_{\text{tar}}) \right]$$

$$= 1 - \left[ \frac{(e)^{-(\rho \cdot \mathcal{A}_{\text{tar}})}(\rho \cdot \mathcal{A}_{\text{tar}})^{(k-1)}}{(k-1)!} + \frac{(e)^{-(\rho \cdot \mathcal{A}_{\text{tar}})}(\rho \cdot \mathcal{A}_{\text{tar}})^{(k-2)}}{(k-2)!} + \cdots \right.$$

$$\left. + \frac{(e)^{-(\rho \cdot \mathcal{A}_{\text{tar}})}(\rho \cdot \mathcal{A}_{\text{tar}})}{1!} + (e)^{-(\rho \cdot \mathcal{A}_{\text{tar}})} \right].$$

$$\eta = 1 - \sum_{h=1}^{k} \frac{e^{-(\rho \cdot \mathcal{A}_{\text{tar}})}(\rho \cdot \mathcal{A}_{\text{tar}})^{(k-h)}}{(k-h)!}. \quad (3-9)$$

➢ *Number of active sensors:* The number of active nodes can be an indicator of network lifetime. Energy consumption is directly depended on this number. Thus, the minimal number with sufficient quality of connected coverage is the goal of selection methods. It is given by:

$$N_a = k \cdot \phi \cdot \omega \cdot \eta, \quad (3-10)$$

where $N_a$ is the total number of active sensors in the network; $k$ is $k$-sensors per target area; $\phi$ is number of target areas per cell (i.e., 1, 3 and 6 for *Centre*, *Lens* and *6-Tri* respectively); and $\omega$ is the number of hexagonal cells which is given by:
In this chapter, we propose a virtual hexagon partition for achieving connected coverage assurance under sensor scheduling. The quality of connected coverage can be examined by solely considering the quality of the coverage.

\[ \omega = \left[ \frac{\mathcal{A}}{\mathcal{A}_{\text{hex}}} \right] = \left[ \frac{2}{3\sqrt{3}} \cdot \frac{W \cdot L}{R_s^2} \right] \]

\[ (3-11) \]

**Quality of the connected coverage:** This is a primary metric that all selection methods have to satisfy the required quality. In fact, this metric consists of two components; connectivity and coverage qualities. However, the quality of connectivity can be guaranteed by the quality of coverage if the relationship between the communication and sensing radii is fixed at \( R_c \geq 2R_s \), according to the Sensing coverage implying connectivity strategy stated in Section 2.3.2 (this condition will be relaxed in the next chapter). Thus, the quality of the connected coverage can be examined by solely considering the quality of the coverage. The quality of the \( k \)-coverage is given by:

\[ \Omega^k = \frac{\sum_{\mu \geq k} 1}{\mathcal{A}} \left( \mu \right) \left( \bigcap_{\mathcal{V}_i} \mathcal{D}_i \right) \]

\[ (3-12) \]

where \( \mu \) is the actual coverage degree in the particular areas for \( \mu = 0, 1, 2, \ldots \); \( \bigcap_{\mathcal{V}_i} \mathcal{D}_i \) is the intersection areas by \( \mu \) sensors (degrees); and \( \bigcup_{\mathcal{V}_i} \left( \bigcap_{\mathcal{V}_i} \mathcal{D}_i \right) \) is all of the areas which are covered by sensors at \( \mu \) degree. The quality of the connected coverage can be evaluated by using a sensing coverage implying connectivity strategy as mentioned in Section 2.3.2. Thus, the quality of \( k \)-coverage implies the quality of the connected coverage.

### 3.5.2 Simulations

The performance of the 6-Tri method is evaluated and compared with the Centre and the Lens [77] methods. The evaluation has been carried out by analysis and simulation in order to validate each other. In the former, the numerical analysis is evaluated using the equations derived above. For the latter, the details are shown in Table 3-1. The values of the network density, sensing and communication radii in the table are set as default, unless stated otherwise. The relationship between both sensing and communication radii is assumed as \( R_c \geq 2R_s \). Since the sensors are randomly deployed into the monitored field, the experiments have been repeated 100 times and the results have been taken from the average values.
Chapter 3. A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

<table>
<thead>
<tr>
<th>HARDWARE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>RAM</td>
</tr>
<tr>
<td>Clock Frequency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SOFTWARE:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>MATLAB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NETWORK SCENARIO:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
</tr>
<tr>
<td>Monitored area</td>
</tr>
<tr>
<td>Base station</td>
</tr>
<tr>
<td>Sensors</td>
</tr>
<tr>
<td>Sensing radius</td>
</tr>
<tr>
<td>Communication radius</td>
</tr>
<tr>
<td>Network density</td>
</tr>
</tbody>
</table>

Figure 3-4 shows clearly that all sensors have an equal chance of being selected by most methods, except the Lens method. In this method, only 42% of the total sensors could be selected, while the rest have no chance of being active. Thus, its virtual partition is needed to recalculate by randomly changing the original position of the partition. Figure 3-5 shows the

![Figure 3-4: Chance of sensor selection ($R_s = 25 \text{ m}$)](image-url)
success of sensor selection from both groups: simulation and analysis by (3-9) when the network density is varied. It is clear that (3-9) is satisfied as the differences between the results of the simulation and analysis throughout the experimental range of the network density are less than 1%. However, for the Centre method, the analytical result is higher than the simulated results by approximately 7% when the network density is around 0.005 #Sensors/m². This is because a small numbers of sensors are deployed, especially throughout area’s boundaries. Consequently, selecting sensors in such hexagonal cells nearby the boundaries is easily failed. In the graphs, the Centre method has the highest rate of success because it considers sensors in a whole hexagonal cell’s area which is greater than the lens and triangle areas. When the number of sensors is sufficient and equally spread, this method is successful for every target area.

The relationship between the total number of active sensors and network density is shown in Figure 3-6. When the sensing radius is fixed at 25 meters, the number of active sensors is increased according to the proportion of selection success. Unfortunately, the 6-Tri method activates more sensors than others because it uses 6 active sensors for covering a cell. Thus, the total number of active sensors is about 4 and 2 times more than the Centre and Lens methods respectively.
Chapter 3. A Virtual Hexagon Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

Furthermore, the sensing radius is a factor which affects the number of active sensors because the quantity of cells in the monitored area is decreased when the sensing radius is increased. As shown in (3-10), the number of active sensors has a number of cells as a parameter. An analysis from this equation is valid as its results differ by only 1% to the simulation’s results. Figure 3-7 shows that the number of active sensors drops very quickly when the sensing radius is increased. At 25 meters of such radius, the total number of active sensors is about 180, 450 and 900 for the Centre, Lens and 6-Tri methods respectively (see also Figure 3-6).
As mentioned previously, the portion between the monitored area and cell’s area affects the number of active sensors. Figure 3-8 shows that their relationship of $k$-coverage where $k$ is equal to 1 and 2 is linear.

Figure 3-9 shows the quality of the connected coverage. In this scenario, the relationship between the communication and sensing radii is $R_c \geq 2R_s$, so the quality of coverage can indicate the quality of connectivity. It is clear that the quality of the connected coverage is satisfied when the network density is sufficient (0.025 #Sensors/m$^2$) for both the Lens and 6-Tri methods, but not for the Centre method, because its quality is only about 0.85. This is because most of the selected sensors are not in the centre of cells, yielding incomplete coverage incurred for those cells. Thus, the Centre method is likely to be critical in terms of the connected coverage guarantee.
Summary

Sensor scheduling is a scheme that can reduce the data load and prolong the network lifetime by activating certain sensors to take responsibility, while others enter sleep mode. However, the quality of service, in terms of sensing coverage and network connectivity, must be controlled at the required level. Therefore, a sensor scheduling method called the 6-Tri method is proposed, which achieves this requirement by utilising a virtual hexagon partition to control the quality. The results show that the quality of connected coverage is maintained; the entire area of the monitored field is fully $k$-covered. Furthermore, all of the sensors across the network have at least one connection with another sensor. In addition, they have an equal chance of being active, so the energy consumption is equally balanced. The analyses are compared to the simulation results in order to validate each other.

However, the 6-Tri method activates more of active sensors in each operational round than the Centre and Lens methods. It is likely that the energy consumption of the 6-Tri method is higher than other methods as the number of active sensors is a main factor. This should be improved further.
Chapter 4

A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

This chapter introduces another sensor scheduling method. This method overcomes the principal problem by using a virtual square partition. The virtual square partition is exploited to partition the sensors into groups of square cells. Representative sensors from each group will be selected to be active and rotated in every operational round. The detail of this sensor scheduling method is presented below.

4.1 Introduction

In Chapter 3, the 6-Tri method was proposed, which has a hexagon tessellation as its virtual partition. Based on the structure of a hexagon, the method picks sensors up from 6 equilateral triangles of a hexagonal cell. Thus, the number of active sensors across the network is given by 6 times the total number of hexagonal cells if the desired coverage degree is equal to 1 \((k = 1)\). Although the 6-Tri method gives complete connected coverage, it seems that it is inefficient in terms of energy consumption.

However, the virtual partition in the 6-Tri method has certain strong properties that can be summarised as:

- All of the sensors in a network have an equal chance of being selected. This is because the chance is given by the ratio of the target areas in which the sensors will be selected to the monitored area. The virtual partition can assign target areas for the whole monitored area.

- The overheads in terms of complexity and message complexity are low. The sensors are physically grouped into cells by the virtual partition. Thus, the sensors are able to perform tasks simultaneously. They are also able to self-organise and locally
negotiate with the members within their group. The information exchange rate is low as the sensors adapt their own information such as position and residual energy.

- The computational cost is low. In the 6-Tri method there are two phases: determining position in the virtual partition, and sensor selecting. To determine the position, the sensors simply compare their own coordinates with those of the BS. For the sensor selecting phase, the sensors use a timer in order to compete to be active within their cell.

It is worth to considering a virtual partition of another shape for a new sensor scheduling method.

4.1.1 Objectives

The objectives of this chapter are as follows:

- To design a sensor scheduling method that uses a virtual partition of a different shape such that it overcomes the principal problem
- To relax the relationship between the communication and sensing radii
- To evaluate the performance and compare the proposed method with others

4.1.2 Contributions

Based on the network model stated in Section 3.2, a sensor scheduling method called the 4-Square (4-Sqr) method is proposed, which controls the connected coverage by using a virtual partition of a square shape. In every scheduling round, the sensors will have an equal chance of being selected from each particular target area, called a sub-square, in the square cells. There are two phases: the initial phase, and the sensor selection phase. In the initial phase, the sensors determine their location on the virtual square partition, based on their own coordinates compared with the base station (BS) at the origin. They are then grouped into square cells and sub-squares, which are the target areas for the sensor selection process. In the sensor selection phase, \( k \) sensors are selected from each target area by considering their own residual energy at the time. The \( k \)-sensors with the highest residual energy will be the first priority to be selected in order to equally distribute the energy consumption among the
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

4.1.3 Chapter organisation

The rest of this chapter starts by understanding the virtual square partition and position realisation in Section 4.2. In Section 4.3, the 4-Sqr method is described, including the process of selecting the sensors and its algorithm. Section 4.4 explains the relaxation of the relationship between the communication and sensing radii. Section 4.5 shows the analyses and results of the simulation. Finally, Section 4.6 summarises the chapter.

4.2 A Virtual Square Partition and Position Determination

This section describes the structure of the virtual square partition in detail. Furthermore, how sensors learn their position on this partition themselves is also explained.

4.2.1 Fundamental square cell and virtual partition

Since a virtual partition consists of number of consecutive cells, the structure of the cells is firstly considered. A fundamental cell is in a square shape whose dimension relates to a sensing disc. Each side of the square is as a chord of the sensing disc. Therefore, the square cell is completely covered by the sensing disc when the centres of both the square cell and the sensing disc are at the same point (see Figure 4-1(a)).
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

Definition 4-1 (Square Cell). The square cell is the fundamental unit of the virtual partition. It is composed of four pieces of the equivalent sub-square, $P_z$ for $z = 0, 1, 2, 3$. Cell’s radius (distance from cell’s centre to one of vertices) is equal to the sensing radius ($R_o = R_s$). The square cell is denoted by $S(O_{uv}, P_z, R_s)$, where $O_{uv}$ is the cell’s centre located at $u$ and $v$ alignments in the virtual partition.

The edge’s length of the square cell is as a chord of the sensing disc given by $2R_s \sin \theta_o = \sqrt{2}R_s$, where $\theta_o$ is the angle diverging for a half of the chord $\left(\frac{\pi}{4}\right)$ subtended at the cell’s centre by the chord. In addition, the edge’s length of the sub-square is half of the cell’s edge i.e. $\frac{R_s}{\sqrt{2}}$. The relationship between the square cell and the sensing disc makes particular parts utilised to describe the coverage property throughout this work. Figure 4-1(b) shows the sector and triangle areas, which are shaded in dark and light grey respectively.

Definition 4-2 (Virtual Square Partition). The virtual square partition is as a square tessellation virtually covering over the monitored area. It is composed of the consecutive square cells, where the original cell is at the base station.
In the virtual square partition, the location of the square cells is indicated by a grid of alignments $u$ and $v$, where $u$ stands for the column on $x$ axis and $v$ stands for the row on $y$ axis (see Figure 4-2(a)). The alignments in the partition are formed from the BS to the network’s edges for $u = (\pm)\{0, 1, 2, ..., U\}$ and $v = (\pm)\{0, 1, 2, ..., V\}$, where $U \leq \frac{W}{R_s \cos \theta_o}$ and $V \leq \frac{L}{R_s \sin \theta_o}$; and $(\pm)$ represents the quadrants of the partition. The coordinates of the square cell, $(x_0, y_0)$ on a monitored area is given by:

$$x_0 = 2u R_s \cos \theta_o, \quad y_0 = 2v R_s \sin \theta_o.$$ (4-1)

### 4.2.2 Position determination

In addition to coordinates $(x_i, y_i)$ on the monitored area, every sensor needs also to determine its position on the virtual square partition. These sensors, then, can be grouped into the particular area which is the sub-square and the square cell. The sensors perform the process of the position realisation only once in the initial phase of the network’s lifetime.

As shown in Figure 4-2(a), the original square cell, $S(O_{00}, P_z, R_s)$, is set at the position of the BS, where $u = 0$ and $v = 0$. Let the coordinates of the original square cell (the BS’s position) on the monitored field be $(x_0 = 0, y_0 = 0)$. Therefore, other square cells are aligned in $x$ axis at $2R_s \cos \theta_o, 4R_s \cos \theta_o, ..., 2u R_s \cos \theta_o$. Meanwhile, they are aligned in $y$ axis at $2R_s \sin \theta_o, 4R_s \sin \theta_o, ..., 2v R_s \sin \theta_o$.

After node deployment, sensors firstly learn their position on a virtual partition by determining the cell’s centre and the sub-square in which they are. Based on coordinate $s(x_i, y_i)$ originated at the BS’s position, every sensor can simultaneously determine lower bound of alignments $\bar{u}$ and $\bar{v}$ by:

$$\bar{u} = \left\lfloor \frac{|x_i|}{(2R_s \cos \theta_o)} \right\rfloor, \quad \bar{v} = \left\lfloor \frac{|y_i|}{(2R_s \sin \theta_o)} \right\rfloor.$$ (4-2)
By a pair of the lower bound alignments from (4-2), the sensor can determine the square cell to which it belongs. There are four possible candidates of the square cells at any point of the sensors in the network (see Figure 4-2(b)):

\[
\begin{align*}
O_a(x, y) &= (\pm)(x_a = \bar{u}2R_s \cos \theta_o, y_a = \bar{v}2R_s \sin \theta_o), \\
O_b(x, y) &= (\pm)(x_b = (\bar{u} + 1)2R_s \cos \theta_o, y_b = 2\bar{v}R_s \sin \theta_o), \\
O_c(x, y) &= (\pm)(x_c = \bar{u}2R_s \cos \theta_o, y_c = (\bar{v} + 1)2R_s \sin \theta_o), \\
O_d(x, y) &= (\pm)(x_d = (\bar{u} + 1)2R_s \cos \theta_o, y_d = (\bar{v} + 1)2R_s \sin \theta_o),
\end{align*}
\]

(4-3)

where \(O_a, O_b, O_c\) and \(O_d\) are the cell candidates and (±) indicates quadrants. From these four candidates, the sensor chooses the closest cell as the associated cell and sets the cell’s position as a reference point. Once its square cell has been realised, the sensor can identify a piece of the sub-square in this cell in which they are exactly laid.

To identify the sub-square, the sensor simply determines a quadrant as there are four quadrants (four pieces of the sub-square) surrounded the recorded reference point (see Figure 4-1). Thus, the piece of the sub-square is given by:

\[
P_z = \begin{cases} 
\mathcal{P}_0 & x_i \geq x_o \text{ and } y_i \geq y_o, \\
\mathcal{P}_1 & x_i < x_o \text{ and } y_i \geq y_o, \\
\mathcal{P}_2 & x_i < x_o \text{ and } y_i < y_o, \\
\mathcal{P}_3 & x_i \geq x_o \text{ and } y_i < y_o.
\end{cases}
\]

(4-4)

Now, the sensor realises its position on the virtual square partition \(S(O_{uv}, \mathcal{P}_z, R_s)\), including the cell’s position \((O_{uv})\) and the piece of the sub-square \(\mathcal{P}_z\). When all sensors have this information, they are ready for a node selection process as described below.

### 4.3 4-Square (4-Sqr) Sensor Scheduling Method

This section explains how sensors are selected from the target areas to be active and how they are scheduled. The aim of this process is to activate certain sensors to operate and others to be put into sleep mode. Meanwhile, the quality of connectivity and sensing coverage is maintained among these active sensors.
4.3.1 4-Sqr method

Four sub-squares in a square cell are used as target areas in which \( k \)-sensors are selected. Thus, this sensor selection is called as the 4-Sqr method. As a square cell consists of four sub-squares, all of the target areas are the entire cell. Therefore, all of the sensors in the network certainly have an equal chance of being selected.

After recognising their position on a virtual square partition, the sensors enter a process of the sensor selection. Based on a square cell, the sensors are grouped into pieces of sub-squares. Only \( k \)-sensors will take responsibility for each piece. Theorem 4.1 shows that each piece of sub-square is required to have active sensors in order to fully cover an entire area of a square cell. Otherwise, the complete connected coverage cannot be guaranteed.

**Theorem 4.1** Any sensor within a piece of sub-squares is able to completely cover this sub-square area. Consequently, a set of sensors within four sub-squares are able to fully cover a whole square cell. Therefore, the complete connected coverage for a whole monitored area is guaranteed if every cell of the square is covered.

**Proof.** The distance between any pair of points within a sub-square area is at most \( R_s \). Hence, any active sensor within a sub-square is able to fully cover this area. However, no sensor can cover the entire area of an adjacent sub-square area. Let \( P_z \) be a piece that has no active sensor, called an empty piece. Thus this empty piece needs to be covered by sensors from both adjacent sub-squares. In the worst case, sensors from both sides are at the furthest vertices which are at corners of the cell in the adjacent sub-squares. Thus, these sensors have to cover the vertex that is the end point of the same edge in the empty piece. According to Definition 4-1, any edge of a square cell is equal to \( \sqrt{2}R_s \). This means that both active sensors from the adjacent sub-squares cannot fully cover the area of the empty piece because they have a sensing radius \( R_s \) which is less than edge’s size \( (\sqrt{2}R_s) \). Therefore, active nodes are needed for each piece of the sub-squares. When there are active sensors for every piece, they can connect to each other since the greatest distance is the diagonal of the square cell, which is \( 2R_s \leq R_c \). Thus, connected coverage is guaranteed. \( \square \)

4.3.2 Processes of the 4-Sqr method

In this section, the processes of the 4-Sqr method are explained. The algorithm has two basic stages: the position determination and the sensor selection. In the former, the sensors perform
the process to obtain their position on the virtual square partition only once in the initial phase. On the other hand, in the latter, the sensor selection process is performed recursively, where the sensors are scheduled. The algorithm starts when node deployment was successful.

In the initial phase, the process is performed only once at the beginning of the network’s lifetime in order to allow sensors to learn their position on the virtual square partition. An outcome of this phase is position on the virtual partition – a square cell and a piece of sub-square on which sensors are laid. A sensor starts with finding lower bound of the horizontal and vertical alignments $\bar{u}$ and $\bar{v}$ by Equation (4-2) before considering cell candidates. There are four possible candidates surrounding the sensor. Then, it determines distances to these candidates. The closet one to the sensor will be the square cell to which it belongs. The centre of this cell is set as a reference point for finding a piece of the sub-square in the next step by Equation (4-4). At the end of the initial phase, all sensors have realised their position on the virtual square partition.

By recorded position on the virtual square partition, every sensor is divided into a group of cells and sub-squares based on their geographical location in a network. Thus, they are now ready to be selected. Only $k$-sensors with the highest level of the residual energy will be active for each group. To achieve that in this distributed system, sensors utilise a timer in the selection competition. The timer has been already given by Equation (3-6) in Section 3.4.3.

### 4.3.3 Sensor selection algorithm

In the algorithm of the 4-Sqr method, every sensor firstly determines its position on the virtual square partition based on the coordinates of the BS (lines 1-4). This process performs only once at the initial phase of the network. The sensors then determine the timer for competition with others. The one with the most residual energy of the group (square cell and sub-square) should be the first node to have its timer expire. The sensor immediately informs its neighbour in the same group and then switches itself to active mode. Other sensors, which have received the announcement, record the new active sensor into a list. The number of the current active sensors must be less than the desired degree ($k$). If the recipients find that the number of the current active sensors in their list has reached the desired degree ($k$), then they terminate their timer and switch to sleep mode (lines 6-25). They will wake up again when the next round comes.
Algorithm: The 4-Sqr Method

Input: \( s(x_i, y_i) \) for \( i = 1, 2, ..., N \), BS(x,y), \( \theta_0 \), \( R_x \), \( k \) and \( E \)

Output: A set of active sensors

Initial Phase:
1. BS broadcasts its coordinates BS(x,y) across the network
2. \( s_1 \) determines its position on the virtual partition \( S(\theta_0, R_x) \) by using (4-2), (4-3), and (4-4)
3. \( e_1 \leftarrow E \)

Begin:
5. Initial_Phase()
6. While \( (e_i \geq \text{given level}) \) /* e.g. \( \geq 50\% \) \( \) or \( 0\% */
7. \( s_i \) determines timer \( t_i \) using (3-6)
8. Waiting \( (t_i) \)
9. If \( (t_i = 0) \) && \( (K < k) \) /* timer is zero and active sensors of this group is less than \( k \) */
10. activeSensor\( (\theta_0, R_x) \) \( \leftarrow s_i \)
11. sendAnnouncement \( (s_i, S(\theta_0, R_x)) \) /* send announcement to neighbours */
12. \( K++ \) /* count current number of active sensors */
13. activeOperating(T) /* this sensor operates in interval T */
14. Elseif \( \) (recAnnouncement \( (s_j, S(\theta_0, R_x)) \) \( = 1 \) /* receive announcement from other sensor */
15. If \( (s_i(S(\theta_0, R_x)) = s_j(S(\theta_0, R_x))) \) /* sender \( j \) is the same group of receiver \( i \) */
16. If \( (K \geq k) \)
17. Terminate \( (t_i) \)
18. GettingSleep \( (s_i) \)
19. Else
20. \( K++ \)
21. resumeWaiting \( (t_i) \)
22. End
23. End
24. End
25. End

4.4 Relaxing Relationship between Communication and Sensing Radii

As the assumption of the relationship between the communication and sensing radii \( (R_c \geq 2R_x) \) is impractical, this relationship here is relaxed to be more realistic. According to Definition 4-1, square cells are formed based on the sensing ability of the sensors, where the radius of the square cells (the distance from cell’s centre to one of the vertices) is equal to the sensing radius \( (R_o = R_x) \), where \( R_o \) is the cell’s radius. Whereas, under the condition that the communication radius is less than two times of the sensing radius, the sensors resize the square cells with the radius equal to half of the communication radius rather than the sensing radius \( (R_o = \frac{R_c}{2}) \). Therefore, the square cell is denoted by \( S(\theta_0, R_c, \frac{R_c}{2}) \). The distance
between adjacent cell’s centres is equal to \( R_c \). Using the sensor scheduling of the 4-Sqr method in Theorem 4.1, the connectivity of sensors between adjacent cells can be maintained.

4.5 Analyses and Simulations

In this section, the performance of the proposed method is evaluated and compared with the Centre, Lens and 6-Tri methods. The network scenario used in this simulation is shown in Table 3-1. There are four metrics to evaluate these methods: the proportion of a chance of sensors being selected, the number of active nodes, the quality of the connected coverage, and the network’s lifetime.

4.5.1 Chance of being selected

This metric has been already explained in Section 3.5.1. The equation of the chance expressed in (3-7) is

\[
\text{Ch} = \frac{\sum \mathcal{A}_{\text{tar}}}{\mathcal{A}} \times 100,
\]

where \( \mathcal{A} \) is the total area of the monitored field, and the target areas \( \mathcal{A}_{\text{tar}} \) are

\[
\mathcal{A}_{\text{tar}} = \begin{cases} 
\frac{3\sqrt{3}}{2} R_s^2, & \text{hexagon area for the Centre method,} \\
\frac{\pi}{3} - \frac{\sqrt{3}}{2} R_s^2, & \text{lens area for the Lens method,} \\
\frac{R_s^2}{2}, & \text{sub-square area for the 4-Sqr method.}
\end{cases}
\]

(4-5)
According to Equation (3-7), the chance of sensors being selected depends on the target areas because only sensors within such areas can be selected. On the other hand, sensors that are outside of the target areas have no chance of being selected at all. Therefore, any selection method considering the entire monitored area as the target areas will have an equal chance for all of the sensors. Figure 4-4 shows that the chance of selecting sensors in the Centre, 6-Tri and 4-Sqr methods is 100 %, while the chance in the Lens method is about 42 %. This is why the authors in [77] need to redraw a virtual partition by generating a random original point for every round of run time.

### 4.5.2 Success of sensor selection

The metric has been already explained in Section 3.5.1, which can be calculated by (3-9). However, the simulation’s result is depicted in Figure 4-4. The tendency of the success in sensor selection carried out by the 4-Sqr method is significantly similar to the 6-Tri method. The success is more than 0.9, although the network’s density is only 0.01 #Sensors/m² and sensors are successfully selected for every target area, when the density is 0.02 #Sensors/m². This means that the 4-Sqr method has potentiality to cover a monitored area.
4.5.3 Number of active sensors

The number of active sensors can be calculated by (3-10) as:

\[ N_a = k \cdot \phi \cdot \omega \cdot \eta, \]

where \( \omega \) is the number of square cells in a monitored area, which is given by:

\[ \omega = \left\lfloor \frac{\mathcal{A}}{\mathcal{A}_{sqr}} \right\rfloor = \left\lfloor \frac{W \cdot L}{2R_s^2} \right\rfloor \]  

(4-6)

The network density is highly influential in the connected coverage. A sufficient quantity of sensors in a network is important for controlling connectivity and coverage and sensor deployment should be equally distributed over a monitored area. Figure 4-5, -5 and -6 show the comparisons of a number of active sensors given by the methods, when the network density, the relationship between the communication and sensing radii, and the sensing radius are varied. Obviously, the number of active sensors becomes steady when the network density is at a sufficient level, which is about 0.025 \#Sensors/m². It is eventually constant regardless of the total number of sensors in the network. In Figure 4-5, the 4-Sqr method activates about 775 sensors, which is less than the 6-Tri method by about 1.2 times, and almost 2 and 4 times more than the Lens and Centre methods, respectively. This is because these methods give a different number of active nodes per cell.
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

Figure 4-6 and Figure 4-7 depict the number of active sensors when the relationship between the communication and sensing radii is varied. In Figure 4-6, the 4-Sqr method activates more sensors when the communication radius is less than the sensing radius in order to maintain connectivity. It can do so by forming the virtual square partition using $R_c^2$ as a cell’s radius. Until the relationship $\frac{R_c}{R_s} \geq 2$ is satisfied, the number of active sensors is stable.

Figure 4-5: Number of active sensors vs. density ($k = 1, R_s = 25 \text{ m.}$)

Figure 4-6: Number of active sensors vs. $R_c/R_s$ ($\rho = 0.025 \text{ #Sensors/m}^2$, $R_s = 25 \text{ m.}, R_c = 50 \text{ m.}$)
Meanwhile, other algorithms cannot maintain the connectivity, so the number of active sensors is likely to be steady. On the other hand, all methods have a very large number of active sensors according to the size of the sensing radius when it is small in order to ensure the quality of sensing coverage. However, the number of active sensors in the 4-Sqr method becomes steady when the relationship is $\frac{R_c}{R_s} < 2$ (25 metres in Figure 4-7) because the connectivity of the sensors is critical. Thus, the 4-Sqr method considers the size of the cell’s radius as equal to $\frac{R_c}{2}$ regardless of the sensing radius.

### 4.5.4 Quality of the connected coverage

There are two qualities in this metric i.e. connectivity and coverage. Unlike Chapter 3, the relationship between communication and sensing radii is relaxed. Thus, the quality of the connected coverage cannot be evaluated by using the *sensing coverage implying connectivity strategy*, which is mentioned in Section 2.3.2. Instead, both coverage and connectivity are evaluated separately, and then the quality of the connected coverage can be evaluated by their results. As mentioned before, every active sensor has to have at least 1 connection to its neighbour. The quality of connectivity is given by:

![Figure 4-7: Number of active sensors vs. sensing radius (\(\rho = 0.025 \text{ Sensors/m}^2\), \(R_c = 50 \text{ m.}\)](image)
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

\[ \Psi = \frac{\sum_{i=1}^{N} a_i}{N_a}, \]

(4-7)

where \( N_a \) is the total number of active sensors, and

\[ a_i = \begin{cases} 1, & \text{if sensor } s_i \text{ has at least 1 connection}, \\ 0, & \text{otherwise}. \end{cases} \]

On the other hand, the quality of \( k \)-coverage is given by:

\[ \Omega^k = \frac{\sum_{\mu \geq k} \bigcup_{\forall i} \left( \bigcap_{\forall i} \mathcal{D}_i \right)}{\mathcal{A}}, \]

(4-8)

where \( \mu \) is the actual coverage degree in the particular areas for \( \mu = 0, 1, 2, \ldots \); \( \bigcap_{\forall i} \mathcal{D}_i \) is the intersection areas by \( \mu \) sensors (degrees); and \( \bigcup_{\forall i} \left( \bigcap_{\forall i} \mathcal{D}_i \right) \) is all of the areas that is covered by sensors at \( \mu \) degree. The quality of the connected coverage is given by:

\[ \Phi = \frac{\Psi + \Omega^k}{2}. \]

(4-9)

For the quality of connectivity shown in Figure 4-8 and Figure 4-9, the 4-Sqr method can obviously maintain complete connectivity for all of the active sensors, regardless of the relationship between the sensing and communication radii. This is because the 4-Sqr method is able to adjust the size of the square cells according to the condition of the relationship as explained in Section 4.4. Thus, distance between a pair of active sensors being in adjacent groups is always less than sensor’s communication radius. This yields good quality not only for connectivity but also coverage as illustrated in Figure 4-11 and Figure 4-12. Meanwhile, the quality of connectivity given by others is suddenly reduced when the sensing radius is greater than half of the communication radius.
Figure 4-8: Quality of connectivity vs. Rs \((\rho = 0.025 \text{ Sensors}/m^2, R_c = 50 \text{ m.})\)

Figure 4-9: Quality of connectivity vs. Re/Rs \((\rho = 0.025 \text{ Sensors}/m^2, R_s = 25 \text{ m.})\)
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

For a part of coverage, most methods are able to guarantee the complete coverage for the whole monitored area when the network has a sufficient density, except the Centre method as shown in Figure 4-10. The Centre method has only 0.85 of the coverage quality due to the error selecting sensors at the cells’ centres. Selecting sensors upon such a target point is very difficult, especially in a random network. Thus, the error can affect the coverage quality of this method. Furthermore, only the 4-Sqr method can maintain the complete coverage, even the ratio of the communication radius over the sensing radius is less than two (see Figure 4-11).

Figure 4-10: Quality of coverage vs. density (\( \rho = 0.025 \) #Sensors/m\(^2\), \( R_s = 25 \) m.)

Figure 4-11: Quality of coverage vs. \( \frac{R_c}{R_s} \) (\( \rho = 0.025 \) #Sensors/m\(^2\), \( R_s = 25 \) m.)
Moreover, the quality of connected coverage is evaluated under positioning error varied from 0 (no error) to 1 (100 metres of error) (see Figure 4-12). Certainly, the 4-Sqr and 6-Tri methods can constantly maintain the complete connected coverage throughout the range of the error. This is because these methods have quite a high number of active nodes and they have the proper size of the target areas (sub-square and triangle for the 4-Sqr and 6-Tri methods).

**Table 4-1: Energy Parameters used in Simulation**

<table>
<thead>
<tr>
<th>Initial Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>Initial energy of a sensor</td>
<td>200 Joules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Energy consumption of a sensor in active and sleep modes (per second)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>Transmission</td>
<td>1400 mW</td>
</tr>
<tr>
<td>Reception</td>
<td>1000 mW</td>
</tr>
<tr>
<td>Idle</td>
<td>830 mW</td>
</tr>
<tr>
<td>Sleep</td>
<td>130 mW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Relevant Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Description</td>
</tr>
<tr>
<td>Bit rate</td>
<td>2 Mbps</td>
</tr>
<tr>
<td>Sampling rate</td>
<td>Every 5 seconds</td>
</tr>
<tr>
<td>Operational interval</td>
<td>5 minutes</td>
</tr>
</tbody>
</table>
methods, respectively). The quality given by the *Lens* method decreases drastically as the sensors with error positions are activated outside the lens areas, so they cannot fully cover their cells.

### 4.5.5 Network’s lifetime

To evaluate the network’s lifetime, the energy consumption measured in [101] is used. The energy parameters of the simulation are detailed in Table 4-1.

Although it consumes the least energy (see Figure 4-15) and has the longest lifetime (see Figure 4-13 and Figure 4-14), the *Centre* method has the worst network stability as certain sensors deplete energy very quickly. As mentioned above, the best quality that this method can achieve is only 0.85, which does not meet our requirement. According to [77], the *Lens* method needs to re-form the virtual partition in order to distribute the chance of sensors being selected. Even though it reveals excellent network stability and quality of connected coverage, the overhead is likely to be high. This can cause the scalability of the method. Obviously, the *4-Sqr* method is better than the *6-Tri* method due to the difference in the number of active nodes. At 50% of the sensors still being alive, the *6-Tri* method reaches 820 seconds, while the *4-Sqr* method can prolong the lifetime to 1150 seconds (see Figure 4-13). The connected coverage is still complete when the time reaches 600 seconds and 900 seconds for the *6-Tri* and *4-Sqr* methods, respectively.

![Network lifetime](image)

Figure 4-13: Network lifetime ($\rho = 0.025 \#\text{Sensors}/m^2, R_s = 25 m.$)
Chapter 4. A Virtual Square Partition for Achieving Connected Coverage Assurance under Sensor Scheduling

4.6 Summary

Sensor scheduling has an important role in limiting the considerable amount of data in networks. However, in scheduling the sensors, the quality of service in terms of connectivity and coverage has to be guaranteed. Therefore, a sensor selection method called the 4-Sqr...
method is proposed, which uses a virtual square partition. In the 4-Sqr method, $k$-sensors are selected to be active from 4 sub-squares per square cell. The 4-Sqre method can achieve all of the properties of the 6-Tri method. Moreover, the 4-Sqr method is more energy efficient than the 6-Tri method. In particular, it relaxes the relationship between the communication and sensing radii. Therefore, the quality of the connectivity and coverage is complete, despite the communication radius being less than the sensing radius.

Although the number of active nodes is likely to be constant regardless of how dense the networks are, the 4-Sqr method activates the sensors more than the Centre and Lens methods. This is because it requires 4 active sensors (if $k = 1$) to cover the entire area of a square cell. This causes a shortened lifetime. Moreover, due to the high number of active sensors, the actual coverage degree is much higher than the desired coverage degree. Thus, the coverage is inefficient. Therefore, these problems need to be addressed in the next chapter.
Chapter 5

Optimisation of Sensor Scheduling Methods for Connected Coverage Assurance

In this chapter, two novel methods for achieving connected coverage assurance under sensor scheduling are presented. Both of them have inherited characteristics from the 6-Tri method, as they exploit a virtual hexagon partition to group the sensors. The first one is the 3-Sym method. It aims to optimise the number of active sensors per hexagonal cell. The other one is the O-Sym method, which enhances the 3-Sym method. It aims to optimise the coverage efficiency to be near to ideal.

5.1 Introduction

According to the main research objectives stated in Chapter 1, the recent methods – the 6-Tri and 4-Sqr methods – have accomplished two purposes: connected coverage assurance and energy balance. The 6-Tri and 4-Sqr methods activate many more sensors per cell than the other methods in each operational round. This is a cause of inefficient coverage and inefficient energy consumption, which is counter to our research objectives. Therefore, in this chapter, the coverage efficiency and the energy efficiency of novel methods are emphasised and improved.

5.1.1 Objectives

The objectives of this chapter are as follows:

- To propose sensor scheduling methods that are able to keep all of the advantages of the recent methods

- To optimise the energy efficiency by minimising the number of active sensors used per cell in order to prolong the network lifetime

- To optimise the coverage efficiency by minimising the overlapping coverage degree to be close to the desired coverage degree
To analyse and compare the performance of the proposed method with other existing methods

5.1.2 Contributions

In this chapter, two novel methods are proposed: the three-symmetrical area method (3-Sym) and the optimum-symmetrical area method (O-Sym). With the excellent structure of a hexagon tessellation, they still exploit the virtual hexagon partition to physically divide the sensors into cells. These methods thus maintain the advantages of the 6-Tri method such as complete connected coverage, low overheads and an equal energy balance.

Based on the virtual hexagonal partition, the question is how many sensors in a hexagonal cell should be selected to be active? Hence, the optimum number of active sensors to fully cover the entire area of a hexagonal cell and to give an equal chance of the sensors in the cell being selected is investigated. It is proven that using three sensors in symmetrical areas is the best solution (this is explained in Section 5.2).

Based on the best solution, the 3-Sym method is proposed. Within each hexagonal cell, the sensor with the most residual energy is promoted to become the first selected sensor: this is called an independent node. Then, the symmetrical areas of the independent sensor are determined and the other sensors in the cell are informed of these. The other two sensors are selected depending on these symmetrical areas and their residual energy. As this process is recursive, the set of active sensors will be rotated in every scheduling round. In this manner, complete connected coverage throughout the monitoring period can be guaranteed. Moreover, the energy consumed by the sensors can be conserved and balanced.

Since the sensor selection process is individually performed in each cell, the overlapping coverage degree is quite high. Consequently, the coverage efficiency and the energy efficiency may be degraded. Thus, the O-Sym method, which is an enhancement of the 3-Sym method, is proposed. The method minimises the overlapping coverage degree to the desired coverage degree: (i) by minimising the overlapping degree and (ii) reducing the proportion of overlapping areas. The O-Sym method considers the actual coverage over a particular area, which is called the sensor's territory, for which the active sensor has responsibility. The actual coverage is made by certain selected sensors around the territory. The method checks the coverage redundancy over the intersection points on the border and inside the territory. The possessor considers releasing its territory by switching to sleep mode, if all of the
intersection points are covered by the other sensors (not those making the examined
intersection point). Therefore, the overlapping coverage degree of this location can be
minimised so that it is close to the desired coverage degree.

Note that our contribution in this chapter is designed based on the network model stated in
Chapter 3.

5.1.3 Chapter organisation

The organisation of the rest of this chapter is as follows. In Section 5.2, the optimum number
of sensors covering the entire area of a hexagonal cell is proved. Then, the protocol according
to the best solution is detailed in Section 5.3. Section 5.4 investigates how to optimise the
coverage efficiency. Section 5.5 presents the analysis and performance evaluation of the
proposed algorithm. Finally, Section 5.6 concludes the chapter.

5.2 Optimum Number of Active Sensors in a Hexagonal Cell

In this section, the optimum number of active sensors to cover an entire hexagonal cell is
investigated. There are three requirements for this investigation: complete connected
coverage, the minimum number of active sensors and an equal opportunity for all of the
sensors to be active. This can be simplified by assigning a coverage degree \( k = 1 \). Hence, the
number of active sensors begins with one node per cell and increases by one until its
properties meet all of the requirements.

5.2.1 One sensor per covering a hexagonal cell

Using one sensor is the lowest number possible for covering a cell. In this case, sensors will
be selected at exact positions such as vertices in the triangle tessellation [99] or centres of the
hexagonal cells in the hexagon tessellation [100]. For the virtual hexagon partition, based on
Definition 3-1 and Definition 3-3, a sensor is able to perfectly cover the entire cell if it is at
the cell’s centre. This sensor selection is called as the Centre method. The quality of the
connected coverage is stated in Theorem 5.1 as follows.

**Theorem 5.1** An active sensor \( s_i \) at the centre of a hexagonal cell is able to cover the entire
area of this cell by its sensing disc, In addition, any pair of active sensors from the adjacent
cells can communicate with each other such that \( R_c \geq 2R_s \). Therefore, the connected
coverage for the whole monitored area is guaranteed if every cell is covered.
**Proof.** Let us firstly consider the coverage upon a piece of the equilateral triangle that is a part of a hexagonal cell. According to Definition 3-3, the radii of both the sensing disc and the hexagonal cell are the same, and thus the area of the triangle is given by $A_{Tri} = \frac{\sqrt{3}}{4}R_s^2$. On the other hand, the sector area of the sensing disc, whose angle subtends the disc’s centre is the same size as the interior angle $\theta_o$ of the hexagonal cell, and is given by $A_{Sec} = \frac{\pi}{6}R_s^2$ (see Figure 3-1(b)). Obviously, the sector area of the sensing disc is greater than the triangle area of the cell, $A_{Sec} > A_{Tri}$. Moreover, every single point in the triangle area is covered by the sector area of the sensing disc, $A_{Tri} \subset A_{Sec}$ if and only if the centre of the cell and the sensing disc are at the same point. Finally, every piece of the hexagonal cell is covered by the sensing disc as the same reason. □

However, the *Centre* method might have the problem of incomplete coverage. According to Theorem 5.1 the centre of the hexagonal cell is only in the right position if the sensor is able to fully cover the cell’s area. Selecting the sensors at the exact positions for every hexagonal cell is impractical, due to the fact that they are randomly deployed into the monitored field. Therefore, the sensors that are the shortest distance from the cell’s centres will be selected instead. For this reason, the quality of coverage cannot be guaranteed due to the error in selecting sensors in the right positions.

### 5.2.2 Two sensors per covering a hexagonal cell

Now, let us consider a two-sensor method. There are two types of used sensors called the *independent* and *dependent* nodes in each hexagonal cell.

**Definition 5-1 (Independent and Dependent Sensors).** The independent sensor is the first selected sensor, which can be anywhere within the cell. It will be the one with the most residual energy. On the other hand, the dependent sensor is the consecutively selected sensor that is selected within a particular area, depending on the independent node’s location, in order to maintain the quality of coverage for the hexagonal cell.

**Definition 5-2 (Critical point on cell’s edge).** The critical point on the cell’s edge is the intersection point, at which a sensor intersects an edge of the hexagonal cell. This point must be covered by another sensor ($k=1$) or $k$ sensors ($k > 1$). The critical point generated by sensor $s_i$ is denoted by $p_c(i)$, and its coordinates are denoted by $(x_c, y_c)$.
The relationship between them in terms of distance and angle is important. Figure 5-1(a) shows an example where the independent node \( s_i \) is selected at an angle equal to \( \pi \). This sensor usually produces two intersection points between its sensing disc and the virtual hexagonal cell. These points are called the critical points \( (p_c(i)) \) because they need to be covered by a dependent node; otherwise the cell is classified as an incompletely covered cell. Therefore, the dependent node \( s_j \) has to be selected in the particular area, depending on the independent node’s position, in order to fully cover the whole cell.

In Figure 5-1(a), the distance \( b \) between the critical points is equal to \( \sqrt{3}R_s \), which is the shortest distance. Thus, a dependent node can be selected from anywhere such that it covers both critical points and the rest of the cell. The dependent node \( s_j \) in Figure 5-1(a) is at the opposite side to the independent one and the distance \( d \) between \( s_i \) and \( s_j \) is equal to \( R_s \). This position of the dependent node is the farthest point for these corresponding angles that full coverage is still maintained. The dependent node can actually be selected at any point from the cell’s centre to \( \frac{R_s}{2} \) with the same angle \( 0 \). Beyond this point, the dependent node cannot maintain full coverage for this cell.

![connected coverage diagram](image)

**Figure 5-1:** Examples of connected coverage by two nodes for a cell (a) the chord that is perpendicular to the hexagonal edges (b) the chord with any angle subtends at the centre of the hexagon (c) the largest chord which is equal to the hexagon’s diameter (d) particular areas where the dependent node can be selected
In fact, the distance \( b \) between the critical points is the chord of the sensing disc, which is the perpendicular distance from both sides of the hexagon. This perpendicular distance relates to the distance \( d \) between the independent and dependent nodes as follows (see Figure 5-1(b)):

\[
d = 2 \sqrt{R_s^2 - \left(\frac{b}{2}\right)^2},
\]

where

\[
b = 2\sqrt{(x_c - x_o)^2 + (y_c - y_o)^2},
\]

\((x_o, y_o)\) and \((x_c, y_c)\) are the coordinates of the cell’s centre and the critical point \( p_c \), respectively.

**Theorem 5.2** Two sensors are able to fully cover the entire area of the hexagonal cell \( \mathcal{H}(O_{uv}, \mathcal{P}_z, R_s) \) if and only if \( d_{i,j} \leq d \), and \( p_c(i) \in D_j \). In addition, both sensors are able to communicate with each another because the distance between them is less than the communication radius \( d_{i,j} \leq R_c \). They also communicate with the other sensors at adjacent cells because the greatest distance among them is less than the communication radius \( \left(2R_s - d_{i,j}\right) \leq R_c \). Therefore, the connected coverage for a whole monitored area is guaranteed if every cell is covered.

**Proof.** A sensor at any location within a hexagonal cell (except at the centre) usually makes two intersection points which occur by its sensing disc crossing the edges of the cell. As the radius of the sensing disc is equal to \( R_s \), the range of the perpendicular distance can vary between \( \sqrt{3}R_s \) as the shortest distance and \( 2R_s \) as the longest distance. Since these points are the critical points, the hexagonal cell can be fully covered such that another sensor covers the rest of it including these critical points. The distance between two sensors can be, at the most, \( d \), when the worst case is \( b = 2R_s \). For full coverage of the cell at this perpendicular distance, the dependent node must be at the centre because it is equal to the disc diameter. \( \Box \)

However, using the *two-sensor* method to cover a cell is likely to incur two problems: incomplete coverage and unequal chance. The former occurs when the independent node does not cover both opposite vertices of the hexagonal cell. Certainly, the distance between
these vertices is the hexagon’s diameter \((2R_s)\). This means that the critical points become the vertices instead of the intersection points made by the independent node. Thus, the right position for a dependent node will be only the cell’s centre (Figure 5-1 (c)). For the same reason as in the Centre method, the coverage cannot be guaranteed due to the rare occasions in which any of the sensors are in the right position.

On the other hand, the sensors may have an unequal chance to become active. Although the independent node can be selected anywhere in the cell’s area, the dependent node is forced to be selected within a limited area. It is always in the black area as shown in Figure 5-1(d). This causes sensors within this area to be active more frequently than others. Therefore, this sensor selection method is not suitable to be used.

### 5.2.3 Three sensors per covering a hexagonal cell

For full connected coverage using three sensors, there is a well-known method in this class, called the Lens method [77]. Based on static virtual partition, the major problem with this method is the unequal chance of the sensors being active. As only sensors within the lens areas will be selected, the others outside of these areas have no chance of being selected at all. Consequently, the network needs to be re-partitioned and the sensors re-grouped for every scheduling round. However, the Lens method has perfect quality of connected coverage based on the symmetrical areas of the three lenses in a hexagonal cell (see Figure 2-16). Therefore, a sensor selection method using three sensors in dynamic symmetrical locations is investigated, to sort out the problem of the Lens method.

This method is to cover a whole cell using three sensors in symmetrical areas. The symmetry is in terms of the angles and distances. The former is the angle between the two sensors and the latter is the distance from the cell’s centre to a sensor. The quantities of both parameters should correspond for all of the sensors in a cell, which represents the symmetry. Lemma 5-1 gives an assurance that a hexagonal cell is completely covered, where the angles and the distances are perfectly symmetrical.

**Lemma 5-1** Let \(\theta_{ij}\) be the symmetrical angle between any pair \(s_i\) and \(s_j\) of three sensors. The angle \(\theta_{ij}\) is a constant equal to \(\frac{2\pi}{3}\). Let \(d_i\) be the symmetrical distance between the cell’s centre and the sensor \(s_i\). All three sensors have the same distance, where \(d_i \leq R_s\). A
A hexagonal cell is guaranteed to be fully covered by the three sensors whose locations are symmetrical.

**Proof.** Unlike the two-sensor method, a dependent node has responsibility for covering about one-third of a cell’s area. In particular, it needs to cover only one of two critical points made by the independent node. Therefore, the worst case is used to prove the relationship between each pair of these three sensors, say \( s_i \) and \( s_j \), where \( d_i = d_j = R_s \) (see Figure 5-2(a)). Thus, the furthest distance between sensors \( s_i \) and \( s_j \) is given by:

\[
d_{i,j} = 2d_i \sin \left( \frac{\theta_{ij}}{2} \right) = 2R_s \sin \left( \frac{\pi}{3} \right) = \sqrt{3}R_s.
\]

Obviously, both sensors \( s_i \) and \( s_j \) intersect at the cell’s centre and point \( q \) due to \( a = c = R_s \). The distance between the cell’s centre and point \( q \) can be calculated as:

\[
b = 2 \sqrt{R_s^2 - \left( \frac{d_{i,j}}{2} \right)^2} = R_s.
\]

As the furthest critical point is a vertex of a hexagonal cell, the greatest distance from the cell’s centre to the critical point is the cell’s radius (see Figure 5-2(b)). Therefore, these two sensors are able to cover this critical point at the cell’s vertex. A hexagonal cell is fully covered by using the perfect symmetry of the three sensors. \( \square \)
As mentioned above in the section on the Centre method, selecting sensors at particular points is a cause of incomplete coverage due to selection errors. Thus, the properties of using three sensors in almost symmetrical positions are considered. This will allow us to select sensors in particular areas rather than at exact points. Lemma 5-2 states that a hexagonal cell is fully covered when three sensors have different distances while their angles are symmetrical. Conversely, Lemma 5-3 states the fact that three sensors with symmetrical distances but different angles are able to cover a hexagonal cell completely.

**Lemma 5-2.** Let $\theta_{ij}$ be the symmetrical angle between any pair $s_i$ and $s_j$ of three sensors, which is equal to $\frac{2\pi}{3}$. Let $d_i$ be any distance from the cell’s centre to $s_i$’s position. A hexagonal cell is fully covered by these three sensors if and only if the differentiation $d_i$ is at most the sensing radius, ($d_i \leq R_s$).

**Proof.** Firstly, a particular area that has to be covered by a pair of sensors $s_i$ and $s_j$ is investigated. It is one-third of a whole cell in a rhombus shape where the length of all of the sides is equal to $R_s$. According to the properties of a rhombus, the opposite sides are parallel, and the opposite angles are equal to $2\theta_0$ and $\theta_0$ (see Figure 5-3). These sensors have symmetrical angles to each other, so $\theta_{ij} = 2\theta_0$. Thus, the sensors are on sides $a$ and $b$ of the rhombus. Let $s_i$ be a sensor at a vertex of the hexagonal cell and $s_j$ be a sensor on the $b$ side at any distance between the cell’s centre and a vertex of the cell (the furthest point).

Henceforth, the challenge is that how both sensors fully cover the rhombus’ area. Precisely, sensor $s_i$ is able to cover the entire area of the equilateral triangle and makes the critical
point \( p_c(i) \) at a vertex of the cell. The rest of the uncovered area is the responsibility of sensor \( s_j \). There are three possible positions for this node, which need to be proved: (I) at the cell’s centre, (II) at a vertex of the cell and (III) at any distance between the previous points. Cases I and II have been already proved according to Theorem 5.1 and Lemma 5.2, which the rest of the cell can be fully covered by sensor \( s_j \). For case III, sensor \( s_j \) can be at any point between the cell’s centre and the vertex, when its angle is fixed at \( 2\theta_0 \). It is able to cover the rest, which is the triangle area as well as the critical point \( p_c(i) \) made by the sensor \( s_i \), \( \{A_{tri} \text{ and } p_c(i) \in D_j\} \). Based on the properties of the rhombus, sensor \( s_j \) always makes the critical point \( p_c(j) \) on the opposite side. Indeed, this point is covered by sensor \( s_i \), \( \{p_c(j) \in D_i\} \). Therefore, both sensors can fully cover the whole area of the rhombus, even though they have different distances from each other. In the same circumstance, the two-thirds of the cell are covered by other pairs of these three sensors.

In another scenario of the almost symmetrical positions, the three sensors have symmetrical distances but different angles. The question here is how much the angles can be different while still maintaining full coverage? For this reason, the greatest angle between the two sensors before losing guaranteed coverage is investigated. Beyond this angle, the coverage will be incomplete. Thus, this angle is called as the critical angle \( \theta_c \) as shown in Figure 5-4(a). Lemma 5-3 gives the conditions of this scenario.

![Figure 5-4: The almost symmetry (symmetrical distances and different angles) of covering a hexagonal cell by three sensors](image-url)
Lemma 5-3. Let $d$ be the distance from the cell’s centre to a sensor, which is symmetrical with the other two sensors in the cell. Let $\theta_{ij}$ be angle between a pair of these sensors. A hexagonal cell is fully covered by these three sensors if and only if the angle between them is at most $2 \cos^{-1} \frac{b^2 + d^2 - R_s^2}{2bd}$, where $b$ is the distance between the critical point and the cell’s centre.

Proof. Let us consider a pair of sensors $s_i$ and $s_j$, where $s_i$ is the independent node and $s_j$ is a dependent node. The independent sensor $s_i$ intersects the cell’s edge at a critical point $p_c$. As the two sensors are the same distance from the cell’s centre ($d_i = d_j$), the angle between them will be the critical angle when $s_i$ and $s_j$ intersect the cell’s edge at the same critical point, $\theta_{ij} = \theta_c$. This obviously makes a graph in a kite shape as shown in Figure 5-4(b).

To find the critical angle $\theta_c$, we firstly determine the critical point $p_c(i)$ which is the intersection point between the sensing disc of sensor $s_i$ and a cell’s edge. In this case as shown in Figure 5-4(b), the cell’s edge is the line $DE$. From this line and the sensing disc of sensor $s_i$ located at $(x_i, y_i)$, we have:

$$y = mx + c \quad \text{and} \quad (x - x_i)^2 + (y - y_i)^2 = R_s^2,$$

(5-1)

where $m$ and $c$ are a slope and the intersection on the $y$ axis of the line, respectively. We then substitute $y$ into the circle equation and reform them into the quadratic formula as shows below [102]:

$$(m^2 + 1)x^2 + 2(mc - my_i - x_i)x + (y_i^2 - R_s^2 + x_i^2 - 2cy_i + c^2) = 0.$$

(5-2)

Thus, $x$ is given by:

$$x = \frac{-(mc - my_i - x_i) \pm \sqrt{(mc - my_i - x_i)^2 - (m^2 + 1)(y_i^2 - R_s^2 + x_i^2 - 2cy_i + c^2)}}{(m^2 + 1)},$$

(5-3)
and \( y \) is given by substituting \( x \) into the line equation in (5-1). Although there will be two sets of coordinates, \((x_1, y_1)\) and \((x_2, y_2)\) from (5-3), only one of these is the critical point, \( p_c(i)\), on the side of the cell such that the distance from this point to the cell’s centre is at most \( R_s\), \( \{(x_c, y_c) | d_{p_c,0} \leq R_s\} \).

To find the critical angle, we firstly calculate the distance \( b \) between the critical point and the cell’s centre (see Figure 5-4(c)). Thus, the angle \( \angle POI \) is given by:

\[
\angle POI = \cos^{-1}\left(\frac{b^2 + d^2 - R_s^2}{2bd}\right).
\]

(5-4)

Based on the properties of a kite graph, the adjacent triangles are congruent, \( \Delta POI \equiv \Delta POJ \). Thus, the diagonal \( PO \) bisects the angle \( \angle O \) which is the critical angle \( \theta_c \). Thus, the critical angle is given by:

\[
\angle O = \theta_c = 2\angle POI = 2\cos^{-1}\left(\frac{b^2 + d^2 - R_s^2}{2bd}\right).
\]

(5-5)

Therefore, complete coverage of a hexagonal cell is maintained as far as the angle between a pair of selected sensors is at most the critical angle.

\[\square\]

From Lemma 5-1, 5-2 and 5-3, we now know that there are certain areas in which three sensors can be selected and cover the entire cell. Hence, the boundaries of these areas can be calculated by determining the point \( (p_p) \) which is the intersection between the perpendicular line \( GP \) and the cell’s edge \( AF \). This point is used as another critical point, which needs to be covered, say \( P \) (see Figure 5-5(a)). Since the line \( GP \) is perpendicular to the line \( HQ \), which is drawn between the critical points made by the independent node and crosses through the cell’s centre, the line \( GP \) is given as:

\[
y = -\frac{1}{m_{HQ}} \times x + \frac{1}{m_{HQ}} \times x_0 + y_0,
\]

(5-6)
where \( O(x, y) \) are the coordinates of the cell’s centre and a slope of the line \( \overline{HQ} \) is given by

\[
m_{\overline{HQ}} = \frac{y_Q - y_H}{x_Q - x_H},
\]

where \( H(x, y) \) and \( Q(x, y) \) are the coordinates of the critical points \( p_c_1(i) \) and \( p_c_2(i) \) made by sensor \( i \) respectively. We also have the cell’s edge \( \overline{AF} \) with which the line \( \overline{GP} \) intersects as:

\[
y = m_{\overline{AF}} x - m_{\overline{AF}} x_A + y_A,
\]

(5-7)

where \( m_{\overline{AF}} \) is a slope of the line \( \overline{AF} \) given by

\[
m_{\overline{AF}} = \frac{y_F - y_A}{x_F - x_A}
\]

and \( A(x, y) \) and \( B(x, y) \) are the coordinates of the vertices A and B respectively. Thus, the coordinates of the intersection point \( P(x, y) \) are given by:

---

Figure 5.5: Three symmetrical areas (a) intersection of critical and perpendicular lines (b) Intersection areas (c) An example of three active sensors within symmetrical areas
\[ x_p = \left( \frac{m_{\overline{AF}} \times x_A + \frac{1}{m_{\overline{HQ}}} \times x_0 + y_0 - y_A}{m_{\overline{AF}} + \frac{1}{m_{\overline{HQ}}}} \right), \]

\[ y_p = m_{\overline{AF}} \times x_p - m_{\overline{AF}} \times x_A + y_A. \]

(5-8)

An equivalent sensing disc \( C(\cdot) \), which has the same property as the sensing disc \( C(\cdot) \equiv D(\cdot) \), is assigned. The equivalent sensing discs are centred at the critical points \( p_{c_1}(i) \), \( p_{c_2}(i) \) and \( p_p \), so the equivalent discs are \( C(p_{c_1}(i), R_s) \), \( C(p_{c_2}(i), R_s) \) and \( C(p_p, R_s) \) as shown in Figure 5-5(b). There are intersection areas between these discs, which are \( C(p_{c_1}(i), R_s) \cap C(p_p, R_s) \) and \( C(p_{c_2}(i), R_s) \cap C(p_p, R_s) \). Indeed, these areas are symmetrical with the position of the independent node and include the particular position stated in Lemma 5-1, 5-2 and 5-3. Therefore, the dependent nodes selected within the symmetrical areas are able to cooperate with the independent node for complete coverage. Theorem 5.3 gives the quality of coverage by using the symmetrical areas.

**Theorem 5.3** Sensors within three symmetrical areas are able to fully cover the entire cell. They are also able to communicate each other within the cell and other adjacent cells. Therefore, the connected coverage for the whole monitored area is guaranteed if every cell is covered.

**Proof.** As \( C(p_{c_1}, R_s) \equiv C(p_{c_2}, R_s) \equiv C(p_p, R_s) \equiv D(s_1, R_s) \), sensors \( s_j \) and \( s_t \), which fall into the intersection areas between the equivalent discs \( C(p_{c_1}, R_s) \cap C(p_p, R_s) \) and \( C(p_{c_2}, R_s) \cap C(p_p, R_s) \), can certainly cover the centres of these discs \( (p_{c_1}, p_{c_2} \text{ and } p_p) \). This is because the distances between the sensors and the centres of the equivalent discs are less than the sensing radius \( \left( d_{j,p_{c_1}}, d_{j,p_p}, d_{t,p_{c_2}}, d_{t,p_p} \leq R_s \right) \). As the covered centres are critical points, the hexagonal cell is fully covered by three sensors \( s_i, s_j \) and \( s_t \) from the symmetrical areas. □

This sensor selection is called as the **Three-Symmetrical Area Method (3-Sym)**. It gives not only full connected coverage for the entire cell but also equal chance of all sensors being active. The intersection areas in which the dependent nodes will be selected randomly change
according to the independent node’s position. With both abilities, full coverage and equal chance, this method utilises only three active sensors per cell, which is considered to be the minimum number to cover the entire cell. The next section will describe this procedure of selecting the sensors in detail.

5.3 Three-Symmetrical Area Method (3-Sym)

This section describes a sensor selection algorithm according to Theorem 5.3 called the Three Symmetrical Area Method (3-Sym). The 3-Sym method uses three active nodes for the desired coverage degree \( k=1 \) (\( 3 \times k \text{ for } k > 1 \)) per hexagonal cell in order to guarantee connected coverage for an entire monitored field. To achieve this, it has two stages in the sensor selection process – independent and dependent selection stages – in order to select an independent node \( (s_I) \) and two other dependent nodes \( (s_D) \). All of the sensors are assumed to be ready for the sensor selection phase, as they have been grouped into a hexagonal cell and equilateral triangle. The process of the position learning phase in the virtual partition has been already described in Section 3.3. The detail of the 3-Sym method is described below.

5.3.1 Independent Sensor Selection

In addition to the equal opportunity for all of the sensors to be active as the major objective, the independent selection will balance the energy consumption across the network. Independent sensors, therefore, are selected based on the condition of their own residual energy. Within each hexagonal cell, the sensor with the most energy decides itself to be the first active (or independent) node of the cell by determining a timer as:

\[
t_I(i) = t_0 \left( \frac{E - \tau e_i}{E} \right).
\]

(5-9)

where \( t_I(i) \) is the timer of the independent selection; \( e_i \) is the residual energy of sensor \( i \); \( E \) is the maximum energy at the beginning; \( \tau \) is a random variable between \([0.9,1]\) to avoid the same value of residual energy from different sensors and \( t_0 \) is a coefficient to limit the waiting time (e.g. \( 1 \times 10^{-6} \) second). According to this timer, the sensor that has sufficient energy will be in charge of sensing and networking onward. The independent node determines the distance \( d_f \) between the cell’s centre and itself, and angle \( \theta_f \) diverging from
the starting angle of each equilateral triangle to one; these are then used as reference points \((p_f)\) for finding the dependent nodes. It then transmits this information to its neighbours within its cell. In addition to the reference points, the information consists of the independent node’s data such as coordinates \(s_i(x, y)\), position in the virtual partition \(\mathcal{H}(O_{uv}, P_z, R_s)\) and residual energy.

The reference distance \(d_f\) and angle \(\theta_f\) can be calculated by (5-10) and (5-11), respectively, based on the coordinates of hexagon’s centre \((x_0, y_0)\) and node \((x_i, y_i)\) as:

\[
d_f = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \tag{5-10}
\]

and

\[
\emptyset = \tan^{-1}\left(\frac{|y_0 - y_i|}{x_0 - x_i}\right), \text{where } x_0 \neq x_i
\]

thus

\[
\theta_f = \begin{cases} 
\emptyset , & \text{if } \left( 0 \leq \theta_i \leq \frac{\pi}{3} \right) \text{ or } \left( \pi < \theta_i \leq \frac{4\pi}{3} \right), \\
\emptyset - \frac{\pi}{3} , & \text{if } \left( \frac{\pi}{3} < \theta_i \leq \frac{\pi}{2} \right) \text{ or } \left( \frac{4\pi}{3} < \theta_i \leq \frac{2\pi}{3} \right), \\
(\pi - \emptyset) - \frac{\pi}{3} , & \text{if } \left( \frac{\pi}{2} < \theta_i \leq \frac{2\pi}{3} \right) \text{ or } \left( \frac{2\pi}{3} < \theta_i \leq \frac{5\pi}{3} \right), \\
(\pi - \emptyset) - \frac{2\pi}{3} , & \text{if } \left( \frac{2\pi}{3} < \theta_i \leq \pi \right) \text{ or } \left( \frac{5\pi}{3} < \theta_i \leq 2\pi \right),
\end{cases}
\]

\[
(5-11)
\]

where \(\theta_i\) is the angle that the independent node \(s_i\) subtends the hexagon’s centre from zero, where \(\theta_i \in [0, 2\pi]\). Note that the angle \(\theta_f\) subtended from the beginning angle of each piece of the equilateral triangles i.e., \(0, \frac{\pi}{3}, \frac{2\pi}{3}, \pi, \frac{4\pi}{3} \text{ and } \frac{5\pi}{3}\); and \(\theta_f \in [0, \frac{\pi}{3}]\). Figure 5-6(a) shows an example in which an independent node \(s_i\) is in the first piece \((P_0)\), so that \(\theta_f = \emptyset\), according to (5-11).
5.3.2 Dependent Sensor Selection

Sensors in the same cell $\mathcal{H}(O_{uv}, P_z, R_s)$ firstly terminate the timer $t_i$, which is in the independent selection process due to the fact there has already been a claim to be the independent node by another sensor. Furthermore, only sensors in the corresponding pieces of the equilateral triangle consecutively determine a reference point symmetrised with $d_f$ and $\theta_f$ for their own piece as well as all of the critical points relating the independent node’s position.

In the dependent node selection, two sensors will be selected as dependent nodes from corresponding triangular pieces. Based on the announced information from the independent node, sensors which are neighbours of $s_i$ are able to determine the coordinates of the symmetrical reference point $p_f(x, y)$ by using (5-12) as follows:
\[
\begin{align*}
    x_f &= x_0 + (d_f \cdot \cos\left(z \cdot \frac{\pi}{3} + \theta_f\right)), \\
    y_f &= y_0 + (d_f \cdot \sin\left(z \cdot \frac{\pi}{3} + \theta_f\right)),
\end{align*}
\]
(5-12)

where \(z\) is the number of equilateral triangles in a hexagonal cell, which is \(z \in 0,1,...,5\). In fact, according to the triangular piece of the independent node, only sensors within corresponding pieces perform the dependent node selection. The corresponding pieces are divided into two groups: \textit{even} and \textit{odd} numbers. In Figure 5-6(b), which shows the \textit{even} case, the independent node \(s_i\) is in \(P_0\), therefore, only sensors within areas \(P_2\) and \(P_4\) are involved in this dependent selection process.

These sensors determine all of the critical points \(p_{c_1}, p_{c_2}\) and \(p_p\) by using (5-3) and (5-8). The sensors consider whether they are able to cover a pair of these points, i.e., \((p_{c_1}, p_p)\) or \((p_{c_2}, p_p)\) (see Figure 5-5). Sensors, say \(s_j\), that satisfy this criterion – \(\left\{d_{j,p_{c_1}} \leq R_s\right\} \text{ and } \left\{d_{j,p_p} \leq R_s\right\}\) or \(\left\{d_{j,p_{c_2}} \leq R_s\right\} \text{ and } \left\{d_{j,p_p} \leq R_s\right\}\) – will calculate a timer for competition in the dependent selection process. Therefore, only sensors that are in particular areas (intersection areas of equivalent sensing discs shown in Figure 5-5(c)) have a chance of being selected for this round.

The timer in the dependent selection process allows the competing sensors to be active without adding the controlling messages too much. The sensor that has the most residual energy and is the closest to the symmetrical reference point will have more chance of being selected than the others. The timer is given by:

\[
t_D(j) = t_0 \left[\alpha \left(\frac{|\theta_f - \theta_j|}{\theta_f}\right) \cdot \left(1 - \frac{R_s - d_{j,p_f}}{R_s}\right) + \beta \left(\frac{|e_i - e_j|}{e_j}\right)\right],
\]
(5-13)

where \(e_i\) is the residual energy of the independent node; \(d_{j,p_f}\) is the distance between a sensor \(s_j\) and the symmetrical reference point \(p_f\); and coefficients \(\alpha\) and \(\beta\) are such that \(\alpha + \beta = 1\). This timer is scaled to a limited waiting time by multiplying by \(t_0\) as mentioned in (5-9). In the first term of (5-13), both the distance \(d_{j,p_f}\), which is compared with \(R_s\) (the maximal distance in the target triangular area), and the angle \(\theta_j\) of sensor \(s_j\) compared with \(\theta_f\) (see
Figure 5-6(b and c)) are considered. The sensor whose timer becomes zero first decides itself to be an active node.

### 5.3.3 Algorithm of Sensor Selection

This section gives the algorithm of the sensor selection process. The output of this process is that the network has certain active sensors that are ready to operate.

<table>
<thead>
<tr>
<th>Algorithm 1: The 3-Sym method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> {s(x_i, y_i)</td>
</tr>
<tr>
<td><strong>Output:</strong> A set of active sensors {s(x_i, y_i)</td>
</tr>
<tr>
<td><strong>InitialPhase():</strong></td>
</tr>
<tr>
<td>1 BS broadcasts its coordinates BS(x, y)</td>
</tr>
<tr>
<td>2 For i = 1, 2, ..., S</td>
</tr>
<tr>
<td>3 (s_i) determines position (H(O_{uv}, P_z, R_s)) [28]</td>
</tr>
<tr>
<td>4 (s_i \leftarrow H(O_{uv}, P_z, R_s))</td>
</tr>
<tr>
<td>5 End</td>
</tr>
<tr>
<td><strong>Independent():</strong></td>
</tr>
<tr>
<td>6 (s_i) determines timer (t_I) by using (5-9)</td>
</tr>
<tr>
<td>7 \text{waitingTimer}(t_I)</td>
</tr>
<tr>
<td>8 If (Timer (t_I) is expired)</td>
</tr>
<tr>
<td>9 (s_i) determines (d_f, \theta_f) by using (5-10) and (5-11) respectively</td>
</tr>
<tr>
<td>10 (s_i \leftarrow s_i)</td>
</tr>
<tr>
<td>11 \text{sendAnnouncement}(Ind_{ID}, e_I, d_f, \theta_f, s(x_i, y_i), H(O_{uv}, P_z, R_s))</td>
</tr>
<tr>
<td>12 Return 1</td>
</tr>
<tr>
<td>13 ElseIf (Timer (t_I) is interrupted by receiving announcement of (s_i))</td>
</tr>
<tr>
<td>14 (s_i) terminates (t_I)</td>
</tr>
<tr>
<td>15 Return 0</td>
</tr>
<tr>
<td>16 End</td>
</tr>
<tr>
<td><strong>Dependent():</strong></td>
</tr>
<tr>
<td>17 (s_i) determines critical points (p_c(s_i)) and (p_p) using (5-3) and (5-8) respectively</td>
</tr>
<tr>
<td>18 If ((d(s_i, p_c(s_i)) \leq R_s) &amp; &amp; d(s_i, p_p \leq R_s)))</td>
</tr>
<tr>
<td>19 (s_i) determines timer (t_D) using (5-13)</td>
</tr>
<tr>
<td>20 Timer (\leftarrow \text{waitingTimer}(t_D))</td>
</tr>
<tr>
<td>21 If (Timer == 1) /* Timer (t_D) is expired */</td>
</tr>
<tr>
<td>22 (\hat{s}_D \leftarrow s_i)</td>
</tr>
<tr>
<td>23 \text{sendAnnouncement}(dep_{ID}, s(x_D, y_D), H(O_{uv}, P_z, R_s))</td>
</tr>
<tr>
<td>24 Return 1</td>
</tr>
<tr>
<td>25 ElseIf ((Timer == 0) &amp; &amp; (P_z(s_D) == P_z(s_i)))</td>
</tr>
<tr>
<td>26 Return 0</td>
</tr>
<tr>
<td>27 End</td>
</tr>
<tr>
<td>28 Else</td>
</tr>
<tr>
<td>29 Return 0</td>
</tr>
<tr>
<td>30 End</td>
</tr>
</tbody>
</table>

---

107
According to Algorithm 1, each sensor primarily learns its position on the virtual partition, based on the coordinates of BS. Consequently, the sensors are grouped into triangular pieces and hexagonal cells (Algorithm1: lines 1-5). Then, the independent node for each cell is selected by a timer, which is utilised by the sensors in the competition. As the value of the timer for selecting the independent node is related to the sensor’s residual energy, competing sensors with the most energy in each cell should be selected. They then inform their neighbours in the cell of the necessary information. On the other hand, because they receive this announcement, the other sensors terminate the timer and move forward into the dependent selection phase (Algorithm1: lines 6-16).

Based on the transmitted information regarding an independent node such as its residual energy and position, the sensors in the corresponding triangular piece (odd or even) determine both of the critical points \((p_c \text{ and } p_p)\) made by the independent node. Only sensors that meet the criteria, which are \(d_{j,p_c} \leq R_s\) and \(d_{j,p_p} \leq R_s\), will determine another timer for the dependent competition. The sensors also determine the reference point that is symmetrical to the position of the independent node. Two sensors from different corresponding triangles, whose timers reach zero first, will be selected to become the dependent nodes. As the timer is a function of position and residual energy, a sensor that is close to the symmetrical reference point and that has a high residual energy will have more chance of being selected than the others (Algorithm1: lines 17-30). The selected sensors from the different triangles inform their neighbours regarding the success of the dependent selection, so that the others will switch to sleep mode until the next round occurs.
5.4 Optimum-Symmetrical Area Method (O-Sym)

This section explains the optimisation of the coverage efficiency after the sensor selection phase. Since the sensors are selected on an individual cell basis using the 3-Sym method, the coverage is inefficient in the global figure because the overlapping coverage degree is too high. The overlapping coverage degree is the degree of the actual intersection that is greater than the desired coverage degree. Thus, this section will explain how to minimise the overlapping coverage degree so as to be close to the desired coverage degree. The section is separated into four parts: the relationship between actual coverage and the desired coverage degrees, the investigation of coverage characteristics, algorithm of the O-Sym method, and relaxation of relationship between sensing and communication radii.

5.4.1 Relationship between actual coverage and desired coverage degrees

The overlapping coverage degree is the one of the important factors that affects coverage efficiency and the lifetime of the WSNs. It implies the inefficient coverage because there are too many active sensors covering and intersecting particular areas in the monitored field. After the sensor selection process, the overlapping coverage degree generally occurs around the monitored area. It should be minimised to the desired coverage degree, so that the coverage can be more efficient. The coverage efficiency (λ) is given by:

\[ \lambda = \mu - k, \]

where \( \mu \) is the actual intersection degree. It could be one of the three cases as:

\[ \lambda = \begin{cases} 
\mu - k > 0, & \text{there exists overlapping,} \\
\mu - k = 0, & \text{efficient coverage,} \\
\mu - k < 0, & \text{incomplete coverage.} 
\end{cases} \]

5.4.2 Investigation of coverage characteristics

The main idea of minimising the overlapping coverage degree is to consider the quality of coverage over a particular area and decide whether or not they need to be active to cooperatively cover such an area. If this particular area is completely covered by other sensors then they switch to sleep mode instead. Based on the distributed networks, each of the active sensors individually considers coverage by its neighbours on its territory. Thus, the
boundary of the territory that an active sensor has responsibility for covering is firstly identified.

- **Territory of an active sensor:** The sensor’s territory is an area where an active sensor has responsibility to cover. It is composed of three adjacent pieces of the equilateral triangles, where the sensor is in the middle piece. Figure 5-7 shows the different boundaries of an active sensor that are in different equilateral triangles. Although the sensing disc of the active sensor may intersect edges of the adjacent triangles of the boundary and may not cover the entire area itself, assigning the explicit territory of an active sensor significantly reduces the complexity of examining the quality of coverage. The definition of a sensor’s territory shows below:

**Definition 5-3 (Sensor’s Territory).** The territory is an area that a sensor $s_i$ is in charge, which is denoted by $T(s_i)$. It is composed of three pieces of adjacent equilateral triangles, where the possessor is in the middle piece. The territory’s area is given by $T(s_i) = 3A_{\text{tri}}$, where $A_{\text{tri}}$ is an area of the equilateral triangle.

- **Characteristics of the coverage on a territory:** As the basic concept of the overlapping coverage minimisation, an active sensor investigates its territory covered by neighbours. The neighbours surrounding the territory are the active sensors, which cooperatively cover a part of or even the entire area of the territory. Therefore, they may be in the same cell or they may be from adjacent cells of the investigating node.
**Definition 5-4 (Neighbours of sensor’s territory).** Let \( T(s_i) \) be a territory of a sensor \( s_i \). A sensor \( s_j \) is a neighbour of the territory of sensor \( s_i \) if there is an intersection area between them, \( T(s_i) \cap D_j \neq \emptyset \). The neighbour of the sensor \( s_i \) is denoted by \( n_j^i \) and a set of neighbours of sensor \( s_i \) is given by \( S_i = \{ n_j^i | i, j \in S_a, j = 1, 2, ..., N_i, i \neq j \} \), where \( S_a \) is a set of active sensors.

In Section 5.2, only one type of critical point on the cells’ edges, which are made by sensors being inside a hexagonal cell, is mentioned. However, in this aspect, covering is performed by active sensors not from only inside but also from outside the cell being investigated, so two different types of critical points are involved. Firstly, the critical point is at the intersection between sensing disc of the active sensor and the boundary of the territory; and secondly, the critical point is at the intersection between the sensing discs of two active sensors, which occurs within the territory.

**Definition 5-5 (Critical point on the territory’s edges).** The critical point on territory’s edge is an intersection point that a sensor intersects an edge of the territory. Let \( p_{cg}^i(s_i) \) be a critical point that the sensor \( s_i \) intersects on its own territory and let \( p_{cg}^i(n_j) \) be a critical point that a neighbour \( n_j \) intersects on an edge of the territory of the sensor \( s_i \). The coordinates of the critical point are denoted by \((x_c, y_c)\).

**Definition 5-6 (Critical point inside the territory).** The critical point inside the territory is an intersection point between two sensing discs occurring within territory’s area. Let \( q_{cg}^i(s_i, n_j) \) be a critical point made by the territory’s possessor \( s_i \) and its neighbour \( n_j \) and let \( q_{cg}^i(n_j, n_l) \) be a critical point between neighbours \( n_j \) and \( n_l \), where \( g \) is the number of the intersection points made by a sensor (in an anti-clockwise). The coordinates of the critical point are denoted by \((x_c, y_c)\).

Figure 5-8 shows both types of critical points. Active sensor \( s_i \) which is in \( P_0 \) is the investigator and has two neighbours \( n_1^i \) and \( n_2^i \). Note that the figure shows only the essential sensing discs in order to avoid confusion. In Figure 5-8(a), the investigator itself produces critical points on the boundaries of the territory, while its neighbours generate two critical points each, i.e., \( p_{cg}^i(n_1) \) and \( p_{cg}^i(n_2) \) by neighbour \( n_1 \) and \( p_{cg}^i(n_2) \) and \( p_{cg}^i(n_2) \) by neighbour \( n_2^i \) (see Figure 5-8(b)). The order of these points of each node is from the first to the last intersections of the sensing disc at the edges of the territory (in an anti-clockwise direction.
from angles 0 to 2\(\pi\). For Figure 5-8(c), the critical point \(q^c_{i}(n_1, n_2)\) is generated by an intersection between the neighbours’ sensing discs. Obviously, only the intersection point occurring inside the territory is taken into account.

- **Investigating quality of the coverage on a territory**: In this section, quality of \(k\)-coverage of the sensor’s territories is investigated. If every single point in this area is \(k\)-covered then the quality is considered to meet the requirement.

A critical point on the territory’s edges indicates the ability of a sensor to cover the territory. An area beyond this point does not belong to this active sensor, so other sensors have to take responsibility, consecutively, in order to maintain \(k\)-coverage over the territory. Otherwise, the territory is incompletely \(k\)-covered. Figure 5-9(a) shows an example in which two critical points produced by \(n^i_1\) and \(n^i_3\) are not covered by any sensors. Thus, this territory is incomplete by \(k\)-coverage (\(k=1\)).

**Lemma 5-4** Any critical point produced on a territory’s edge by an active sensor has to be covered by other \(k\) sensors in order to maintain \(k\)-coverage. Otherwise, the \(k\)-coverage of the territory is incomplete.
Chapter 5. Optimisation of Sensor Scheduling Methods for Connected Coverage Assurance

Proof. It is usual that at least one active sensor has produced critical points on the edges of the territory after finishing the sensor selection process. This means that this sensor covers a part of the territory, \( \mathcal{D}_i \cap \mathcal{T}(s_i) < \mathcal{T}(s_i) \). As a critical point represents the finite ability of a sensor to cover on the territory, the actual coverage degree beyond this point decreases by one \((\mu - 1)\). If the rest of the actual coverage degree is less than the desired coverage degree, \( \mu - 1 < k \), then another sensor must cover over this critical point in order to compensate for the falling degree. In other words, there are \( k \) sensors covering this critical point \( p^i_k(s_i) \in \bigcap_{j=1}^k \mathcal{D}(n_j, R_s) \), otherwise the coverage is incomplete by \( k \)-coverage. \( \square \)

However, although every critical point on the edges of the territory has been \( k \)-covered, this cannot guarantees that the territory is fully \( k \)-covered. There is another type of critical point that occurs at the intersection between the sensing discs inside the territory, which needs to be considered. Figure 5-9(b) shows an example where the critical points are not covered. This is a cause of incomplete coverage as shown in the black area. Thus, these critical points also need to be covered.

Lemma 5-5 Any critical point produced by an intersection between two sensing discs has to be covered by other \( k \) sensors in order to maintain the \( k \)-coverage. Otherwise, the \( k \)-coverage of the territory is incomplete.

Proof. In cooperative coverage by a certain number of active sensors, the critical points between the sensing discs may be produced inside the territory. Indeed, the critical point shows the bounds of coverage by a pair of active sensors. Precisely, the covered area by two
sensors is less than the territory, \( (D(n_j, R_s) \cup D(n_{j+1}, R_s)) \cap T(s_i) < T(s_i) \), and the actual coverage degree over the particular area beyond this point decreases by one \((\mu - 1)\). If the rest of the actual coverage degree is less than the desired coverage degree, \( \mu - 1 < k \), then another sensor must cover this point in order to compensate for the decreased degree. Therefore, the critical point is \( k \)-covered when \( q_c^k(s_i, n_j) \in \cap_{l=1}^k D(n_l, R_s) \) and \( i \neq j \neq l \).

From Lemma 5-4 and 5-5, the property of the coverage over the territory could be stated by Theorem 5.4.

**Theorem 5.4** If every critical point produced within the territory is \( k \)-covered then the entire territory is completely \( k \)-covered.

**Proof.** As each territory is cooperatively covered by certain active sensors, it is normal that there are critical points of both types – inside and on an edge of a cell – appeared within the territory except for in these two cases: only one active sensor is at the cell’s centre or there is no sensor surrounding the territory with a sensing radius. In the former, there is no any intersection point occurred within the territory or even cell’s area. It is actually the minimum coverage degree for this cell already. On the other hand, the circumstance in the latter occurs when the network has either a lack of deployed sensors or a lack of living sensors due to depleted energy. Hence, the coverage degree of these cases cannot be optimised. Obviously, the coverage optimisation can be performed when critical points occurred within the territory, using both Lemma 5-4 and 5-5.

Base on this investigation, the algorithm is provided, including detail how to optimise quality of the coverage successfully in such a distributed network in the next section.

### 5.4.3 Algorithm of the O-Sym method

The optimisation is to minimise the number of active sensors, while full coverage is guaranteed for the entire monitored field. In such distributed networks, a decentralised operation of sensors is considered for optimising the overlapping degree. Each of the active sensors investigates the covering circumstances over its own territory. If it finds that the territory is completely covered by the others without involving itself, then it negotiates with its neighbours in order to release the territory by switching to sleep mode. Each of the active sensors performs the optimisation process as in the algorithm shows below:
### Algorithm 2: The O-Sym method

**Input:** \( s(x_i, y_i), N = \{n_j^i| j = 1, 2, ..., N_i\} \) and \( \mathcal{H}(O_{uv}, P_s, R_s) \)  
**Output:** A set of active sensors optimised.

**Begin:**  
1. \( N = 0, P = 0, Q = 0 \)  
2. \( Pc \leftarrow \text{InvestigationPc}(N, P) \)  
3. If (\( Pc = 1 \)) /* all critical points are covered */  
4. \( Qc \leftarrow \text{InvestigationQc}(N, Q) \)  
5. If (\( Qc = 1 \)) /* all critical points are covered */  
6. \( Ne \leftarrow \text{Negotiation}(N) \)  
7. If (\( Ne = 1 \)) /* negotiation is successful */  
8. \( s_i \) switches to sleep mode  
9. Else  
10. \( s_i \) is still active  
11. End  
12. Else  
13. \( s_i \) is still active  
14. End  
15. Else  
16. \( s_i \) is still active  
17. End  

**End**

**InvestigationPc**(\( N, P \));  
18. \( s_i \) picks up \( n_j \) which has \( \max_{1 \leq j \leq N} (e_{res}(j)) \)  
19. \( a = 1 /* index of set \( N */ \)  
20. \( b = 1 /* index of matrix \( P */ \)  
21. \( N_1 \leftarrow n_j /* adding to a set of associated neighbours */ \)  
22. While 1  
23. \( s_i \) determines \( p_{bg}^i(n_j) \) for all \( g \) where \( 1 \leq g \leq 4 \) by using (5.2) and (5.3)  
24. \( P_{bg} \leftarrow \bigcap_{g=1}^4 (p_{bg}^i(n_j)) /* adding the critical points of \( j \) into the matrix \( P */ \)  
25. \( P_{bg} \leftarrow 0 /* marking all new critical points as uncovered */ \)  
26. For \( l = 1, 2, ..., N_i \) and \( j \neq l \)  
27. For \( g = 1, 2, 3 \) and \( 4 /* there are at most 4 critical points */ \)  
28. If \( d\left(p_{bg}^i(n_j), n_l\right) \leq R_s \)  
29. \( P_{bg} \leftarrow 1 /* marking this point is covered */ \)  
30. If (\( n_l \notin N \))  
31. \( a = a + 1 /* increasing index by 1 */ \)  
32. \( N_1 \leftarrow n_l /* adding to a set of associated neighbours */ \)  
33. End  
34. End  
35. End  
36. If (\( P_{bg} = 1 \)) /* all critical points made by \( n_j \) covered */  
37. break!  
38. End  
39. End  
40. If (\( b < a \)) /* if there is more sensors to consider */  
41. \( b = b + 1 /* increasing index */ \)  
42. \( n_j \leftarrow N_b /* the next sensor of associated neighbours */ \)  
43. ElseIf (\( P = 1 \)) /* all element in \( P \) is equal to 1 */  
44. Return 1 /* critical points \( p_c \) is completely covered */  
45. Else  
46. Return 0 /* critical points \( p_c \) is incompletely covered */
Investigation\text{Qc}(\mathbb{N}, \mathbb{Q}):\n\text{For } j = 1, 2, \ldots, |\mathbb{N}|, \n\text{For } l = 1, 2, \ldots, |\mathbb{N}|, \n\text{If } (j == l)\n\text{\quad } Q_{jl} \leftarrow \text{`-'} /* \text{None intersection here */} \n\text{Else}\n\quad s_i \text{ determines } q^l_i(n_j, n_l) \text{ by using (5-15) to (5-18)} \n\text{\quad } Q_{jl} \leftarrow q^l_i(n_j, n_l) = 0 \text{ and/or } /* \text{one or both points is(are) */} \n\text{\quad } Q_{lj} \leftarrow q^l_j(n_j, n_l) = 0 /* \text{in the territory */} \n\text{For } a = 1, 2, \ldots, N_i \text{ where } a \neq j \text{ and } a \neq l \n\text{\quad } \text{If } (q^l_i(n_j, n_l) \notin \mathcal{D}(n_a)) \n\text{\quad \quad } Q_{jl} \leftarrow 1 \text{ and/or} \n\text{\quad \quad } Q_{lj} \leftarrow 1 \n\text{\quad \quad } \text{If } (n_a \notin \mathbb{N}) \n\text{\quad \quad \quad } \mathbb{N} \leftarrow n_a \n\text{End}
\text{End}
\text{End}
\text{End}
\text{End}

\text{Negotiation}(\mathbb{N}): \n\text{sendReq } (\mathbb{N}) /* s_i \text{ sends `Request’ to all associated sensors in the set } \mathbb{N} */ \n\text{Req = 1 /* flag of sending request*/} \n\text{Ack \leftarrow receiveReq(Req)} \n\text{If } (\text{Ack} == 1) \n\text{\quad \quad \quad \quad \quad \quad \quad } \text{Return 1} \n\text{Else} \n\text{\quad \quad \quad \quad \quad \quad \quad } \text{Return 0} \n\text{End}

\text{receiveReq(Req)}: \n\text{If } (\text{Req} != 1) \n\quad s_i \text{ stops determining } p^l_i(\cdot) \text{ and } q^l_i(\cdot) \n\quad \text{sendAck(sender) \leftarrow 1} \n\text{Else} \n\quad \text{If } (e_{\text{res}}(i) > e_{\text{res}}(j)) /* \text{where } i \text{ is receiver and } j \text{ is sender */} \n\quad \quad s_i \text{ stops determining } p^l_i(\cdot) \text{ and } q^l_i(\cdot) \n\quad \quad \text{sendAck(sender) \leftarrow 1} \n\quad \text{Else} \n\quad \quad \text{sendAck(sender) \leftarrow 0} \n\text{End}
As this process is performed after the sensor selection process, all of the active sensors assume that they already have their neighbours’ information such as their coordinates and residual energy, through local announcement when they become active. Thus, each active sensor $s_i$ has a set of neighbours $S_i = \{n_j^i | i, j \in S_a, j = 1,2,...,N_i, i \neq j\}$. Based on this information, all of the sensors simultaneously determine the critical points on the territory’s edges $p_i^c(\cdot)$ by using line-circle intersection equations as shown in (5-2) and (5-3). Furthermore, they determine the critical points $q_i^c(\cdot)$ occurring because of the intersection between the two sensors by using circle-circle intersection equations. From the sensing discs of two sensors $s_i$ and $s_j$, we have:

\[
(x - x_i)^2 + (y - y_i)^2 = R_i^2, \quad (5-15)
\]
\[
(x - x_j)^2 + (y - y_j)^2 = R_j^2. \quad (5-16)
\]

We expand and subtract them and then we have $x$ as:

\[
x = \frac{(x_i^2 - x_j^2) + (y_i^2 - y_j^2) - 2(y_i - y_j)y}{2(x_i - x_j)}. \quad (5-17)
\]

We rewrite (5-17) as $x = \omega - \sigma y$, where

\[
\omega = \frac{(x_i^2 - x_j^2) + (y_i^2 - y_j^2)}{2(x_i - x_j)} \quad \text{and} \quad \sigma = \frac{(y_i - y_j)}{(x_i - x_j)}.
\]

Substituting $x$ into (5-15) and expanding it, we have

\[
(\sigma + 1)y^2 + 2(\sigma x_i - \sigma - y_i)y + x_i^2 + y_i^2 - 2\omega x_i + \omega^2 - R_i^2 = 0. \quad (5-18)
\]
From (5-18) we can find y and then substitute this into either (5-15) or (5-16) to retrieve the coordinates of the two intersection points [103]. However, we are interested only in those that are within the territory of sensor $s_i$.

In this process, every sensor has three stages to carry out: investigating the covering on the critical points on the territory’s edges, investigating the covering on the critical points of the sensing disc intersection and negotiating with its associated neighbours. If any one of these is not successful then the investigating sensor will stop the optimisation process and decide to stay in active mode (lines 1-17). To balance energy consumption, the active sensor $s_i$, firstly chooses its neighbour $n_j$ with the most residual energy and determines the critical points $p_i^c(n_j)$ produced by this node on the possible territory edges. The number of critical points is at the most 4, and these are kept in a matrix $\mathbb{P}$:

$$\mathbb{P} = \begin{pmatrix}
p_{11} & p_{12} & p_{13} & p_{14} \\
p_{21} & p_{22} & p_{23} & p_{24} \\
\vdots & \vdots & \vdots & \vdots \\
p_{N_i1} & p_{N_i2} & p_{N_i3} & p_{N_i4}
\end{pmatrix},$$

where the columns represent the points and the rows represent the neighbours. The sensor $s_i$ then finds out whether or not these critical points are covered by other neighbours. If it finds any neighbour covering one or more critical points then it determines the critical points produced by this node. This process will run recursively until the sensor $s_i$ finds that all of the critical points are covered. Cooperatively covering neighbours are recorded into a set of associated nodes $\mathbb{N}$ where $\{\mathbb{N} \subseteq N_i | N_i$ is a set of the neighbours of sensor $s_i\}$ (lines 18 - 48).

In investigating the critical points $q_i^c(\cdot)$, sensor $s_i$ determines the intersection points between a pair from the set of associated nodes $\mathbb{N}$ and keeps records of them in a matrix $\mathbb{Q}$:

$$\mathbb{Q} = \begin{pmatrix}
q_{11} & q_{12} & \cdots & q_{1N_i} \\
q_{21} & q_{22} & \cdots & q_{2N_i} \\
\vdots & \vdots & \ddots & \vdots \\
q_{N_i1} & q_{N_i2} & \cdots & q_{N_iN_i}
\end{pmatrix},$$

where both the columns and rows represent neighbours. Normally, a pair of sensors produces two intersection points, but only points within the territory are critical points. The active
sensor $s_i$ finds out the covering over these critical points by others from the set of neighbours. The covering must satisfy the condition in Lemma 5-5. The process will run recursively until every point has been considered. If all critical points are covered then the active sensor $s_i$ negotiates with associated nodes (lines 49-76).

After checking the quality of coverage, this node sends a request message to its associated neighbours and waits for an acknowledgement. The receivers immediately terminate the optimisation process and send an acknowledgement to cooperatively cover the territory of sensor $s_i$. In the case that the requesting sensor receives a request message from another, the decision will be made based on the residual energy between them (lines 77-93). The active sensor $s_i$ changes to the sleep mode if the negotiation is successful.

### 5.4.4 Relaxation of Relationship between Sensing and Communication Radii

As the assumption of the relationship between sensing and communication radii ($R_c \geq 2R_s$) of sensors is not valid all the time, therefore this relationship is relaxed to be close to the reality. When the sensors have the ability to communicate with others less than two times of their sensing ability ($R_c < 2R_s$), they must form the virtual hexagon partition with the radius of the hexagonal cell equal to $R_h = \frac{R_c}{2}$ instead of being the sensing radius. By this cell’s radius, the connectivity between sensors can be guaranteed regardless of the sensing radius as the furthest distance within the cell (at the opposite vertices) is equal to $R_c$. Therefore, the hexagonal cell is denoted as $\mathcal{H}(O_{uv}, P_z, R_h)$, and the coordinates of the cell from Definition 3-3 becomes:

$$x_O = 3uR_h \cos \theta_O,$$

$$y_O = \begin{cases} 
2vR_h \sin \theta_O, & \text{if } u \text{ is even}, \\
(2v + 1)R_h \sin \theta_O, & \text{if } u \text{ is odd},
\end{cases}$$

Note that the detail of how the sensors obtain the coordinates of their own cell is described in Section 3.3.

### 5.5 Analyses and Simulations

This section presents the analyses and evaluations of the proposed methods compared with other existing methods. In addition, the key factors that indicate the methods’ performance,
such as the chance of sensors being selected, the quality of connected coverage, the number of active sensors and the coverage efficiency, are evaluated.

Here, performance of the methods introduced in this chapter – 3-Sym and O-Sym – is compared with the Centre, 2-Sensor, and Lens methods, which are in One-, Two- and Three-sensor categories, respectively as explained in Section 5.2. Furthermore, the 6-Tri method from Chapter 3 as well as the CCP [78] and OGDC [104] are involved in this evaluation.

The network scenario used in this simulation is shown in Table 3-1. Furthermore, the energy parameters of the simulation are detailed in Table 4-1. The results are as follows.

### 5.5.1 Time complexity

The 3-Sym and O-Sym algorithms are absolutely decentralised and self-organised. In the former, its time complexity is composed of two elements. The first element is from the initial phase in which sensors individually learn their position on the virtual hexagon partition based on BS’s coordinates. The time complexity of this phase is constant $O(1)$. The second element is from the sensor selection phase in which an independent and other two dependent sensors are selected from each hexagonal cell. The sensor selection phase is driven by both timers $T_I$ and $T_D$. The time complexity of this element is given by $O(max(T_I + T_D))$. It can be accounted as constant $O(1)$.

On the other hand, in addition to the time complexity of the 3-Sym algorithm, the O-Sym algorithm has extra time complexity for the coverage efficiency optimisation phase. In this phase, each sensor simultaneously considers overlapping on its territory. The time complexity could be $N_i^3$, where $N_i$ is the number of neighbours surrounding the territory of sensor $s_i$. Fortunately, the number of neighbours is constant as there are three selected sensors of each hexagonal cell. Thus, the maximum number of neighbours is $N_i = 20$ (3 nodes × 6 cells + 2 nodes from the same cell). Therefore, its time complexity is constant $O(1)$. 


5.5.2 Chance of Sensors Being Selected

Usually, sensors have a chance of being selected as they lie in the target areas. Hence, the chance is the proportion of the target areas over the entire monitored field. As only the selected sensors consume energy, the sensors having an equal chance involves balancing the energy consumption across the network. The chance is given by Equations (3-7) and (3-8).

In Figure 5-10, most of the methods reveal the equal chance that sensors have of being selected, except for the 2-Sensor and Lens methods, which has only a 70% and 42% chance, respectively. Obviously, the main reason affecting the chance of the deployed sensors is the proportion of target areas over the monitored area. For the 2-Sensor and Lens methods, their target areas are isolated from others, for example lens areas, so that sensors deployed outside of the target area have no chance of being selected. Thus, the proportion of target areas is less than the monitored field's area. Meanwhile, the 6-Tri and 3-Sym methods have equilateral triangles and symmetric areas, respectively, as their target areas. All of these target areas is equal to the entire monitored area. Therefore, all of the sensors in the monitored field have an equal chance of being selected.

5.5.3 Success of Sensor Selection

The success of the selection equates to the proportion of the sensors that are successfully selected from the target areas. It is given by Equation (3-9)
Figure 5-11 demonstrates how sensors are successfully selected when the network density varies. In other words, the number of sensors is required by any selection method. The success, in fact, indicates the quality of the connected coverage. This quality can be guaranteed if the sensors are successfully selected from the target areas. It is clear that the success of the sensor selection directly relates to the sizes of the target areas. To successfully select the sensors in the particular areas, the Centre method requires the network density only 0.01 #Sensors/m², which is similar with the 2-Sensor method. Meanwhile, the 6-Tri and 3-Sym methods require a network density about 0.02 #Sensors/m². The Lens method is quite sensitive to the network density as it has the smallest target area, so it easily fails to select the sensors in such target areas. Therefore it requires a very high density of sensor deployment about 0.05 #Sensors/m².

5.5.4 Number of Active Sensors

After the sensor selection phase, selected sensors decide to be in active mode, while unselected sensors switch to sleep mode. The active sensors are now on behalf of the sleep sensors for their own region and are in charge of performing not only sensing but also data delivery tasks. The number of active sensors, therefore, is a primary factor indicating the network’s life. the number of active sensors can be determined by Equation (3-10).

The number of active sensors becomes steady when the network density is sufficient (see Figure 5-12). As it uses only one sensor to cover the entire cell, the Centre method seems to
be an ideal in this aspect because it gives the minimum number of active sensors. Meanwhile, the 2-Sensor method in Two-sensor category and the 3-Sym method in the Three-sensor category (see Section 5.2) activates about 370 and 480 sensors, respectively, when the sensor selection is successful. However, the O-Sym method significantly minimises the number of active sensors less than the 3-Sym, OGDC and CCP algorithms about 30%, 17% and 10% respectively, where the network density is 0.025 #Sensors/m².

Furthermore, the number of active sensors also varies according to the relationship of the communication radius over the sensing radius \( \frac{R_c}{R_s} \) as depicted in Figure 5-13 and Figure 5-14. In general, a network with a very small sensing radius scale will have a very high number of sensors activated; conversely, the number of active sensors is decreased dramatically when the sensing radius’s scale increases. In Figure 5-13, the numbers of active sensors of the Centre and 3-Sym algorithms decrease continuously according to the scale of the sensing radius. However, the relationship between communication and sensing radii cannot be neglected, as shown in the results of the O-Sym, CCP-SPAN and OGDC algorithms. The numbers of active sensors of these algorithms become steady when the communication radius is less than two times of the sensing radius (\( R_c < 2R_s \)). Under this scenario, the O-Sym method forms the hexagonal cells based on the communication radius instead, \( \left( R_h = \frac{R_c}{2} \right) \), according to Section 5.4.4. Since the communication is fixed (\( R_c \) is set to 50 meters), the number of active sensors becomes steady as the sensing radius is greater than 25 meters.
Similarly, the CCP with a SPAN feature and OGDC algorithms have an algorithm that can maintain connectivity among sensors, yielding the constant number of active sensors regardless of whether or not the sensing radius has been increased.

Similar to the reasons mentioned above, the numbers of active sensors of these algorithms will increase drastically when the communication radius is less than two times of the sensing radius or $\frac{R_c}{R_s} < 2$ (see Figure 5-14) as they activate more sensors in order to guarantee the quality of the connectivity. The numbers rather become steady when this proportion meets
the criterion \( \frac{R_c}{R_s} \geq 2 \). In contrast, the Centre and 3-Sym algorithms have a constant number of active sensors regardless of the proportion.

### 5.5.5 Connectivity and Coverage

Connected coverage is the primary metric whose quality must be satisfied. The connectivity of an active sensor satisfies if and only if there is a link and one can communicate with another, according to Definition 3-2. Quality of connectivity is therefore the proportion of connected sensors and unconnected sensors, which is given by Equation (4-7). On the other hand, the quality of coverage can be examined from the proportion of the covered areas over the entire monitored field. According to Definition 3-1, the quality of coverage is considered as satisfactory if the monitored field is fully covered with at least \( k \) degrees. The proportion of the coverage by at least \( k \) degree is given by Equation (4-8). Furthermore, the quality of the connected coverage is given by Equation (4-9).

The graph in Figure 5-15 illustrates the quality of connected coverage compared to the network density. Only the Centre method cannot have full connected coverage even the network density is very high due to errors in selecting sensors on the correct locations as mentioned in Section 5.2. Meanwhile, other methods can reach fully connected-cover rapidly even if the network density is very low, e.g. a density of 0.008 #Sensors/m\(^2\). This indicates a tolerance for missing active sensors for each method. In addition, Figure 5-16 shows the

![Quality of connected coverage vs. density](image_url)
quality of connectivity, when the sensing radius is changed. The quality of connectivity that the Centre and 3-Sym methods have drops rapidly when the sensing radius is greater than 25 metres (more than half of the communication radius). Furthermore, in the Centre method, active sensors cannot communicate with any one of them when the sensing radius reaches 40 metres. Thus, this method is the most sensitive with regard to this issue, while other algorithms can maintain the connectivity among sensors. The O-Sym method has excellent quality as it can maintain connectivity of active sensors by re-partitioning the deployed sensors with the communication radius rather than the sensing radius.

The quality of connected coverage can also suffer because of the position error of the deployed sensors as shows in Figure 5-17. Both of our methods cannot guarantee the complete coverage if the rate of the position error is more than 10 metres. Their quality gradually decreases to 0.98 (or a decrease of 1.46%) and 0.97 (2.4%) for the 3-Sym and O-Sym methods respectively, when the rate of the position error is 50 metres. However, our algorithms are better than the OGDC and CCP algorithms whose quality starts to decrease as the error of positioning reaches only 5 metres. At this error rate, their quality dramatically decreases to 0.95 (4.1%) and 0.93 (6.7%), respectively. This indicates how much the areas are uncovered when the sensors have a position error. For instance, as the monitored area is set to 500x500 m², uncovered areas are about 3,666 m², 6,041 m², 10,344 m² and 16,807 m² for the 3-Sym, O-Sym, OGDC and CCP algorithms respectively, when the position error is 50 metres.
Chapter 5. Optimisation of Sensor Scheduling Methods for Connected Coverage Assurance

5.5.6 Coverage Efficiency

According to Equation (5-14), the coverage efficiency ($\lambda$) is the relationship between the actual coverage degree ($\mu$) and the desired coverage degree ($k$). The proportion of coverage efficiency is given by:

$$\Lambda^\lambda = \frac{\bigcup \left\{ \bigcap \mathcal{D}_i \right\}}{\mathcal{A}}$$

(5-19)

where $\lambda \in \{-k, 1-k, \ldots, -2, -1, 0, 1, 2, \ldots, \mu - k\}$. The coverage efficiency shows the elements of the coverage proportion in each degree incurred in the monitored field. In case of $\lambda = -k$, this element presents the proportion of the coverage hole, which certain areas are not covered by any sensor. Meanwhile, in the case of $\lambda = 0$, this is the proportion of the perfect coverage efficiency because the actual coverage degree is equal to the desired coverage degree. Indeed, when $\lambda \geq 1$, there are overlapping coverage degree incurred in certain areas.

In the result of the proportion of coverage efficiency (Figure 5-18), the 3-Sym method has the greatest proportion of overlapping coverage degree at 2 and 3, which is inefficient. Thus, the proportion can be optimised by reducing this overlapping coverage degree using an algorithm stated in Section 5.4.3. The optimised 3-Sym method called the O-Sym method significantly reduces the overlapping degree making it closer to the desired coverage degree ($\lambda = 0$). It
gives the proportion of the overlapping degrees from 2 and 3 in the 3-Sym method to 1 and 2 as the highest ones, which is difficult to avoid in a case of complete coverage by circular discs. Even though the Centre method has excellent proportion at an efficient coverage degree, unfortunately are incompletely $k$-covered ($\lambda < 0$). This does not satisfy the requirement.

Moreover, the O-Sym method minimises the number of active sensors less than the 3-Sym method about 30%, where the network density is 0.02 #Sensors/m$^2$ (see Figure 5-18). This indeed can conserve a considerable amount of energy. Although the O-Sym method has about 45% more active sensors than the Centre method the quality of coverage is different; therefore the Centre method needs additional number of active sensors to improve this quality.

### 5.5.7 Network lifetime and network stability

In this section, the algorithms in terms of the lifetime and stability of the network are evaluated. The parameters of energy used in the simulation of this section are expressed in Table 4-1.
There are two factors that potentially affect the network’s lifetime and stability. The first factor is the number of active sensors which indicates the quantity of energy consumption. The second factor is the sensors’ load which is any task that sensors perform when they are active. Figure 5-19 depicts the lifetime and stability of the network that the algorithms can achieve. The O-Sym algorithm obviously gives the network the longest lifetime because not only does it have the least number of sensors activated, but also it can balance the sensor’s load quite well by rotating duties among the sensors. Figure 5-20 shows the mean of the energy consumption with standard deviation. The standard deviation of our algorithms is at
most 15 Joules spread out from the mean, which is smaller than the OGDC and CCP which have 22 and 50 Joules, respectively. This is one reason why both OGDC and CCP algorithms’ lifetime is not very different to the lifetime achieved by the 3-Sym algorithm, despite the fact that they have significant differences in the number of active sensors. The stability of the CCP algorithm is even worse than others because it has no a mechanism to balance the sensor’s load. 

Figure 5-21 illustrates the quality of the connected coverage throughout the network’s lifetime. It is clear that the O-Sym algorithm can maintain complete coverage for the entire monitored field for 1800 seconds, even the number of sensors which are alive decreases to as little as 93%. Meanwhile, the OGDC and CCP algorithms can only maintain the same quality when the lifetime reaches 1130 and 1090 seconds, respectively. However, the OGDC and CCP can tolerate the high rate that the sensor’s energy depletes as they provide a high level of quality of connected coverage, even though the number of sensors which are alive is low. For example, when 50% of the sensors are alive, the CCP can maintain the quality of connected coverage at 0.9. For the same quality, the OGDC can maintain connected coverage whilst only 30% of the sensors are alive in the network.

5.6 Summary

In this chapter, two sensor scheduling methods – 3-Sym and O-Sym – are proposed. They use a virtual hexagon partition. The 3-Sym method selects and activates certain sensors within
symmetrical areas of each hexagonal cell, while the other sensors are put into the sleep mode. Based on this sensor selection, not only can complete connected coverage be guaranteed but also all of the sensors have an equal chance of being selected, and the number of active sensors is the optimum for every scheduling round. Moreover, the overlapping degree incurred by the 3-Sym method can be significantly reduced by applying the O-Sym method. This method allows the active sensors to consider the coverage upon their own territory. They release their territory and switch to the sleep mode if their territory has already been fully covered by their neighbours. Therefore, the coverage efficiency is optimised by reducing the overlapping coverage degree to close to the desired coverage degree and the number of active sensors is also minimised. Therefore, all of the research objectives are accomplished by the O-Sym method.
Chapter 6

Conclusions and Future Works

6.1 Conclusions

One of ultimate goals of wireless sensor networks (WSNs) is a long lifetime. By the nature of WSNs, the energy resources of sensors are limited and it is impractical to either recharge or replace them as the sensors are often deployed in large and inaccessible areas. Therefore, efficient energy consumption by the sensors needs to be approached. One of the most effective methodologies is sensor scheduling.

The sensors, which are randomly and densely deployed in a monitored area, can be scheduled in order to conserve the energy consumed by them. In a sensor scheduling methodology, the operational time of a network can be divided into rounds of a certain interval. In each round, a certain number of selected sensors are active and perform their responsibilities such as sensing and transmitting data. In the meantime, some of the other sensors simply enter into sleep mode, which consumes much less energy. However, scheduling the sensors has a potential impact on the quality of service in terms of sensing coverage and network connectivity. Therefore, the quality of sensing coverage and network connectivity (known as the quality of connected coverage) must be maintained at a satisfactory level, whilst the sensors are scheduled.

From the family of connected coverage problems mentioned in Section 2.1, the entire area connected coverage problem (EACC) is addressed. To overcome this problem, it is necessary to design a mechanism that is able to guarantee the quality of connected coverage for the entire monitored area, and simultaneously maximise the network’s lifetime, despite the sensors being scheduled. Therefore a series of sensor scheduling methods is proposed, namely 6-Tri, 4-Sqr, 3-Sym and O-Sym, which achieve both the required quality of connected coverage and the prolongation of the network lifetime.

All of the methods control the quality of connected coverage using a virtual partition in order to divide the sensors into groups of cells and even sub-groups within a cell, according to the
sensors’ locations in the monitored field. A cell is defined as a single unit of a virtual partition, which can be either a hexagon or a square. The idea of grouping the sensors is to assign a finite area within which members are responsible for full coverage, and to allow members to connect with others in the adjacent groups.

The sensor scheduling methods have been designed based on a decentralised manner. Thus, in the initial phase, every sensor simultaneously determines its position in the virtual partition and identifies a group to which it belongs. The information obtained includes the cell’s alignments ($O_{uv}$) and a sub-group ($P_z$) in the cell. The overheads for this phase are very low as the time complexity is constant ($O(1)$) and the message complexity is only one message broadcast by the BS. Later on, sensors can be selected to be active by different methods as summarised below.

Firstly, a sensor scheduling method called the 6-Tri method is introduced. It uses a hexagon tessellation as a virtual partition over a monitored field. This partition divides the sensors into groups of hexagonal cells and sub-groups of triangular pieces (there are 6 triangular pieces in a hexagonal cell), based on the sensors’ locations. A hexagonal cell and its triangular piece are denoted by $\mathcal{H}(O_{uv}, P_z, R_s)$, where $O_{uv}$ is the cell’s centre along with its alignments, $P_z$ is a triangular piece, and $R_s$ is the radius of the cell, which is equal to the sensing radius. The 6-Tri method selects $k$ sensors with the highest residual energy from each triangular piece. The method uses a timer as a strategy to select the sensors. Every sensor determines a timer based on how much residual energy it has. The value of the timer is an inverse function of the energy amount, so that $k$ sensors with the highest residual energy in a group will have the shortest timer to count down and will become active on behalf of the group.

The 6-Tri method controls the quality of connectivity between sensors using the implying strategy as mentioned in Section 2.3.2. The communication radius is assumed to be at least twice of the sensing radius ($R_c \geq 2R_s$). Thus, connectivity can be guaranteed because the virtual hexagon partition is constructed based on the size of the sensing radius. In addition, the 6-Tri method controls the quality of coverage by having $k$ active sensors in each triangular piece. As a sensor can fully cover the entire area of a triangle, a hexagonal cell will be fully covered if there is an active sensor from each triangle ($6$ active sensors per cell). Likewise, $k$ sensors from each triangular piece give $k$-coverage for the entire area of a hexagonal cell. Theorem 3.2 proves that the quality of connected coverage is guaranteed for the entire area of the monitored field, if every hexagonal cell is fully $k$-covered.
The main achievements of the 6-Tri method are as follows:

- The required quality for both sensing coverage and network connectivity is achieved. The required quality of coverage is such that the entire area of the monitored area is fully \( k \)-covered. On the other hand, the required quality of connectivity is such that every active sensor is connected to another.

- The energy consumption is equally balanced among the sensors. This is achieved by two factors: the chance of sensors being selected and the sensor rotation. With regard to the former, the chance of being selected is given by the ratio of the target areas (triangular pieces) to the monitored area. Thus, every sensor has an equal chance of being selected regardless where it is.

- The overheads in terms of time complexity and message complexity are very low. The time complexity is constant, and is related to the value of the timer. On the other hand, the message complexity is linear as \( O(N_a) \), where \( N_a \) is the number of active sensors.

Secondly, the question then arises, “Is there any other shape of partition that can control the quality of the connected coverage while the sensors are being scheduled?” Hence, the 4-Sqr method is introduced to answer the question. The 4-Sqr method uses a square tessellation as a virtual partition. Based on the structure of the partition, the sensors are grouped into square cells and sub-squares. Note that there are 4 sub-squares inside a cell. A square cell and sub-square can be denoted by \( S(O_{uv}, P_z, R_s) \). Similar to the 6-Tri method, \( k \) sensors are selected from each sub-square in order to fully \( k \)-cover the entire area of a square cell. Therefore, the quality of connected coverage is guaranteed if every cell is \( k \)-covered as stated and proved in Theorem 4.1.

Unlike the 6-Tri method, in addition to the implying strategy used to control the quality of connectivity, the 4-Sqr method relaxes a constraint regarding the relationship between the sensing and communication radii. In the case of the communication radius being less than twice of the sensing radius (\( R_c < 2R_s \)), the method resizes the radius of the square cells, so that it is equal to half of the communication radius instead of the sensing radius (\( R_o = \frac{R_c}{2} \)), where \( R_o \) is the cell’s radius. Therefore, the square cell can be rewritten as \( S(O_{uv}, P_z, \frac{R_c}{2}) \). Due to this relaxation, the connectivity between active sensors can be controlled.
The main achievements of the 4-Sqr method are as follows:

- All of the advantages produced by the 6-Tri method remain in the 4-Sqr method.
- The 4-Sqr method activates sensors less than the 6-Tri method by about 20%, so it has a longer network lifetime. For example, when 50% of the sensors are still alive, the 4-Sqr method can operate for 1,150 seconds, whilst the 6-Tri method reaches only 820 seconds.
- The 4-Sqr method relaxes the assumption regarding the communication and sensing relationship. Thus, it is more practical than the 6-Tri method.

Thirdly, the 3-Sym method is proposed in order to answer the question, “How energy consumption can be further reduced?” Due to the excellent structure, the virtual hexagon partition, which is used by the 6-Tri method, is used. The major factor in reducing the energy consumption is the number of active sensors. Therefore, the optimisation of the number of active sensors per hexagonal cell is initially studied. Three active sensors per hexagonal cell is the optimum number as expressed in Section 5.2.

The 3-Sym method is designed based on the optimum solution. Its procedure has two phases: independent sensor selection and dependent sensor selection. In the first phase, a sensor is selected independently within the cell’s area, based on the residual energy. The 3-Sym method then selects another two sensors from particular areas, which are symmetric with the position of the independent sensor. As the 3-Sym method is de-centralised, these processes are performed in each cell simultaneously. Thus, the time complexity is equal to both timers for the independent \( T_I \) and dependent \( T_D \) sensor selection. It is given by \( O(\max(T_I + T_D)) \), which is constant. On the other hand, the message complexity is linear \( O(N_a) \).

The main achievements of the 3-Sym method are:

- All of the advantages produced by the 6-Tri and the 4-Sqr methods remain in the 3-Sym method
- The 3-Sym method can significantly minimise the number of active sensors to a greater extent than the 6-Tri and 4-Sqr methods. For example, when the density is equal to 0.025 #Sensors/m², the numbers of active sensors that are activated by the 6-Tri
Tri, 4-Sqr and 3-Sym methods are 900, 780 and 480, respectively. Therefore, the 3-Sym method has a much longer network lifetime than the others.

Finally, the overlapping coverage degree produced by the 3-Sym method is optimised by a novel sensor scheduling method called O-Sym to achieve efficient coverage which is one of main research objectives. As the 3-Sym method activates 3 sensors per cell, the degree of actual coverage generated by this method is higher than the required degree \( k \). The O-Sym method has the 3-Sym algorithm as its sensor selection state. Thus, a set of sensors are activated from every hexagonal cell. Each active sensor takes responsibility for investigating the coverage redundancy over its territory (see Figure 5-7).

**Theorem 5.4** reveals a concept for checking the complete \( k \)-coverage over a sensor’s territory. The \( k \)-coverage can be proved by considering the intersection points occurring within a territory. The intersection points are made by nearby active sensors whose sensing discs intersect on the territory’s boundary and intersect with each other. The \( k \)-coverage of the entire area of the territory is considered to be complete if every intersection point is \( k \)-covered. Based on this theorem, the O-Sym method simply lets the active sensors check the coverage redundancy within their own territory. If an active sensor finds that its territory is fully \( k \)-covered by its neighbours without being active itself then the active sensor is eligible to release its territory and enter into sleep mode. Therefore, the coverage redundancy can be precisely minimised in particular areas.

Although the O-Sym method requires extra process compared to the 3-Sym method and the time complexity should be \( N_i^3 \), where \( N_i \) is the number of neighbours surrounding the territory of sensor \( s_i \), the number of neighbours, in fact, is constant, as there are, at the most 3 active sensors per cell and there are only 6 cells surrounding the sensor’s territory. Therefore, the time complexity is constant \( O(1) \). The message complexity is linear \( O(N_a) \) based on the number of active sensors.

The main advantages of the O-Sym method are as follows:

- All of the advantages produced by the 6-Tri, 4-Sqr, and 3-Sym method remain in the O-Sym method
Chapter 6. Conclusions and Future Works

- The O-Sym method significantly minimises coverage redundancy. The actual coverage degree is optimised to converge on the desired coverage degree for the whole monitored area.

- As the coverage redundancy is minimised, the number of active sensors is also reduced and consequently the network lifetime is prolonged.

The series of sensor scheduling methods proposed has achieved all of main research objectives, as stated in Section 1.3. Furthermore, it has a high rate of tolerance over position error of the sensors. Using a virtual partition, all of the methods have very low overheads in terms of time complexity and message complexity. This thesis gives comprehensive detail and novel methods for connected coverage assurance under sensor scheduling.

6.2 Future Works

Future works that could be put forward are as follows:

- It is more realistic if the probabilistic sensing coverage model (see Section 2.2.2) is applied instead of the sensing disc model. The quality of coverage in term of coverage probability is likely to become a major issue. Obviously, this factor can impact many aspects such as algorithms to maintain quality of coverage and connectivity, the number of active sensors and the coverage efficiency.

- It is a challenge to control the quality of sensing coverage and network connectivity, particularly in a heterogeneous network. In such a scenario, the sensors’ capabilities are different. In particular, the sensing and communication radii are two major factors that directly affect the quality guarantee when the sensors are scheduled. A controlling algorithm must realise the coverage range as well as the communication range of the sensors individually. In particular, it must deal with the quality of sensing coverage and the quality of connectivity separately.

- The mobility of sensors also poses challenges in controlling the sensing coverage and connectivity. A sensor’s movement can cause problems such as a coverage hole or even a network partition. Thus, one challenge is how to guarantee complete coverage over the target areas and guarantee a connected network for all time. In future works, an algorithm could check the quality of connected coverage in real-time while a
sensor is being scheduled. Furthermore, the algorithm might exploit the potentiality of mobility to overcome the coverage hole and network partition problems.

- Three-Dimension (3-D) network scenarios bring incredible challenges for the connected coverage control problem. There are numerous applications in 3-D networks, for example, underwater acoustic monitoring [105], aerosphere pollution monitoring [106], and even monitoring on complex surfaces such as volcanos [107]. The existing strategies for controlling the quality of connected coverage for two-dimension (2-D) planes are not workable in 3-D scenarios. Controlling both the sensing coverage and network connectivity in a 3-D network is much more difficult than in a 2-D plane.
Appendix A

Programming Code of the 6-Tri Method

This section provides the programming code of the 6-Tri method, including the hexagon partition, and the sensor selection by the 6-Tri method.

A.1 Hexagon partition

### TABLE A–1: HEXAGON PARTITION

```matlab
Main (){  
  % The structure of variables  
  % n(x,y,hx,hy,piece)  
  %The structure of sn is sn(x,y,hx,hy,piece,energy,modeFlag,freq)  

  N = round((density*wide*length)/(pi*Rs^2));  
  % u = (N*pi*R^2)/A  
  for j=1:1:100  % repeat for 100 times  
    n = 0; sn = 0;  
    n = Dep(N,area);  % sensors are deployed randomly into the network.  
    n = PosH(n,Rs);  % finding sensors’ position and grouping sensors  
  in hexagon partition  
    sn = Sort(n);  % sorting  
    sn = A3_6Pcs(sn,wide,length,Rs);  % running sensor selection with  
the 6-Tri method  
    end  
  
end

function [n] = Dep(N,wide,length)
  %%%%%%%%%Nodes deployment uniquely and randomly%%%%%%%%%%%%%%%%
  n = random('Unif',-wide/2,length/2,2,N);
end

function [n] = PosH(n,Rs)
  %Finding nodes’ position  
  N = size(n,2);  
  slope = 1.73;  
  for i=1:1:N  
    a = 1; b = 1;  
    s = 0; f = 0;  
    x = n(1,i);  
    y = n(2,i);  
    if (x < 0)  %quadrant of nodes  
      a = -1;  
      end  
    if (y < 0)  
      b = -1;  
      end
```
Appendix A: Programming Code of the 6-Tri Method

s = floor(abs(x)/(3*Rs*cosd(60))));  % find the floor on x axis
f = floor(abs(y)/(Rs*sind(60))));    % find the floor on y axis

%%%%% Finding candidates of hexagons %%%%%%%%%%%%%%%
if (mod(s,2) == mod(f,2))
    candidate(1,1) = a*s*3*Rs*cosd(60);
    candidate(2,1) = b*f*Rs*sind(60);
    candidate(1,2) = a*(s+1)*3*Rs*cosd(60);
    candidate(2,2) = b*(f+1)*Rs*sind(60);
elseif (mod(s,2) ~= 0)
    candidate(1,1) = a*s*3*Rs*cosd(60);
    candidate(2,1) = b*(f+1)*Rs*sind(60);
    candidate(1,2) = a*(s+1)*3*Rs*cosd(60);
    candidate(2,2) = b*f*Rs*sind(60);
elseif (mod(f,2) ~= 0)
    candidate(1,1) = a*(s+1)*3*Rs*cosd(60);
    candidate(2,1) = b*f*Rs*sind(60);
    candidate(1,2) = a*s*3*Rs*cosd(60);
    candidate(2,2) = b*(f+1)*Rs*sind(60);
end

%%%%%%%%%% Finding the closet hexagon %%%%%%%%%%%
if (sqrt((x-candidate(1,1))^2 + (y-candidate(2,1))^2)) <= (sqrt((x-
    candidate(1,2))^2 + (y-candidate(2,2))^2))
    n(3,i) = candidate(1,1);
    n(4,i) = candidate(2,1);
else
    n(3,i) = candidate(1,2);
    n(4,i) = candidate(2,2);
end

%%%%%%%%%%%% Finding quadrant %%%%%%%%%%%%
r = 0;  % radius of the node from centre of hexagon
d = 0;  % the opposite side of angle
A = 0;  % the angle
hx = n(3,i);  hy = n(4,i);
if (x >= hx) && (y >= hy)  % Q1
    n(5,i) = floor(atand(abs(y-hy)/abs(x-hx))/60);
elseif (x < hx) && (y >= hy)  % Q2
    n(5,i) = floor((180-atand(abs(y-hy)/abs(x-hx))))/60);
elseif (x < hx) && (y < hy)  % Q3
    n(5,i) = floor((180+atand(abs(y-hy)/abs(x-hx))))/60);
else  % Q4
    n(5,i) = floor((360-atand(abs(y-hy)/abs(x-hx))))/60);
end
end
A.2 Sensor selection of the 6-Tri method

### Table A–2: Sensor Selection of the 4-Sqr Method

```matlab
function [ sn ] = A3_6Pcs( sn,wide,length,Rs )
%The Algorithm #3: 6 Pieces in a Hexagon
N = size(sn,2);
sn(7,:) = 0;
p0 = 0; p1 = 0; p2 = 0; p3 = 0; p4 = 0; p5 = 0;
for i=1:1:N
    if (sn(9,i) ~= 0)
        switch (sn(5,i))
            case 0
                sn(7,i) = 1;
                p0 = i;
            case 1
                sn(7,i) = 1;
                p1 = i;
            case 2
                sn(7,i) = 1;
                p2 = i;
            case 3
                sn(7,i) = 1;
                p3 = i;
            case 4
                sn(7,i) = 1;
                p4 = i;
            case 5
                sn(7,i) = 1;
                p5 = i;
        end
    elseif (sn(3,i) == sn(3,i-1)) && (sn(4,i) == sn(4,i-1))
        switch (sn(5,i))
            case 0
                if (p0 == 0)
                    sn(7,i) = 1;
                    p0 = i;
                elseif(sn(6,p0) < sn(6,i))
                    sn(7,i) = 1; % activate
                    sn(7,p0) = 0; % reset
                    p0 = i; % keeping
                end
            case 1
                if (p1 == 0)
                    sn(7,i) = 1;
                    p1 = i;
                elseif(sn(6,p1) < sn(6,i))
                    sn(7,i) = 1; % activate
                    sn(7,p1) = 0; % reset
                    p1 = i; % keeping
                end
            case 2
                if (p2 == 0)
                    sn(7,i) = 1;
                    p2 = i;
                elseif(sn(6,p2) < sn(6,i))
                    sn(7,i) = 1; % activate
                    sn(7,p2) = 0; % reset
                end
        end
    end
end
```

141
Appendix A: Programming Code of the 6-Tri Method

```plaintext
p2 = i; % keeping
end

case 3
if (p3 == 0)
    sn(7,i) = 1;
p3 = i;
elseif(sn(6,p3) < sn(6,i))
    sn(7,i) = 1; % activate
    sn(7,p3) = 0; % reset
    p3 = i; % keeping
end

case 4
if (p4 == 0)
    sn(7,i) = 1;
p4 = i;
elseif(sn(6,p4) < sn(6,i))
    sn(7,i) = 1; % activate
    sn(7,p4) = 0; % reset
    p4 = i; % keeping
end

case 5
if (p5 == 0)
    sn(7,i) = 1;
p5 = i;
elseif(sn(6,p5) < sn(6,i))
    sn(7,i) = 1; % activate
    sn(7,p5) = 0; % reset
    p5 = i; % keeping
end

else
    switch (sn(5,i))
    case 0
        sn(7,i) = 1;
p0 = i;
p1=0; p2=0; p3=0; p4=0; p5=0;
    case 1
        sn(7,i) = 1;
p1 = i;
p0=0; p2=0; p3=0; p4=0; p5=0;
    case 2
        sn(7,i) = 1;
p2 = i;
p0=0; p1=0; p3=0; p4=0; p5=0;
    case 3
        sn(7,i) = 1;
p3 = i;
p0=0; p1=0; p2=0; p4=0; p5=0;
    case 4
        sn(7,i) = 1;
p4 = i;
p0=0; p1=0; p2=0; p3=0; p5=0;
    case 5
        sn(7,i) = 1;
p5 = i;
p0=0; p1=0; p2=0; p3=0; p4=0;
end
end
end
end
```
Appendix B

Programming Code of the 4-Sqr Method

This section provides the programming code of the 4-Sqr method, including the square partition, and sensor selection by the 4-Sqr method.

B.1 Square partition

```
function [n] = Dep( N,area )
    %Nodes deployment uniquely and randomly
    n = random('Unif',-area/2,area/2,2,N);
end

function [ nSq ] = PosSq( nSq,Rs )
    %The Function of finding nodes' position for square virtual partition
    N = size(nSq,2);
    for i=1:N
        a = 1; b = 1;
        s = 0; f = 0; % s:x exist, f:y exist
        x = nSq(1,i);
        y = nSq(2,i);
        if (x < 0) %quadrant of nodes
            a = -1;
        end
        if (y < 0)
```

**TABLE B–1: SQUARE PARTITION**

```plaintext
Main (){ % The structure of variables
% nSq(x,y,hx,hy,peice)
% snSq(x,y,hx,hy,piece,energy,modeFlag,freq)

N = round((density*wide*length)/(pi*Rs^2)); % u = (N*pi*R^2)/A
for j=1:1:100 % repeat for 100 times
    nSq = 0; snSq = 0;
    nSq = Dep(node,area); %sensors are deployed randomly into the
    nSq = PosSq(nSq,Rs); % finding sensors' position and grouping
    sensors in hexagon partition
    snSq = Sort(nSq); % sorting
    snSq = A4_4Sqr(snSq,wide,length,Rs); % running sensor selection
    with the 4-Sqr method
end
}
```
b = -1;
end
s = floor(abs(x)/(2*Rs*cosd(45))); % find the floor on x axis
f = floor(abs(y)/(2*Rs*sind(45))); % find the floor on y axis

%%%%% Finding candidates of square %%%%%%%%
candidate(1,1) = a*s*2*Rs*cosd(45); %
candidate(2,1) = b*f*2*Rs*sind(45); % c3 c4
candidate(1,2) = a*(s+1)*2*Rs*cosd(45); % x
candidate(2,2) = b*f*2*Rs*sind(45); % c1 c2
candidate(1,3) = a*(s)*2*Rs*cosd(45);
candidate(2,3) = b*(f+1)*2*Rs*sind(45);
candidate(1,4) = a*(s+1)*2*Rs*cosd(45);
candidate(2,4) = b*(f+1)*2*Rs*sind(45);

%%%%%%%%%% Finding the closet square %%%%%%%%%%%
dt = 0;
st = 0; %the shortest distance at moment
for c=1:1:4
  dt = sqrt((x-candidate(1,c))^2 + (y-candidate(2,c))^2);
  if (c == 1)
    st = dt;
    nSq(3,i) = candidate(1,c);
    nSq(4,i) = candidate(2,c);
  elseif (st > dt)
    nSq(3,i) = candidate(1,c);
    nSq(4,i) = candidate(2,c);
    st = dt;
  end
end

%%%%%%%%%%%% Finding quadrant %%%%%%%%%%%%
r = 0; % radius of the node from centre of square
d = 0; % the opposite side of angle
A = 0; % the angle
sx = nSq(3,i); sy = nSq(4,i);
if (x >= sx) && (y >= sy) % Q1
  nSq(5,i) = 1;
elseif (x < sx) && (y >= sy) %Q2
  nSq(5,i) = 2;
elseif (x < sx) && (y < sy) %Q3
  nSq(5,i) = 3;
else %Q4
  nSq(5,i) = 4;
end
B.2 Sensor selection of the 4-Sqr method

<table>
<thead>
<tr>
<th>TABLE B–2: SENSOR SELECTION OF THE 4-SQR METHOD</th>
</tr>
</thead>
</table>

```matlab
function [ sn ] = A4_4Sqr( sn,wide,length,Rs )
    %The finding successful selected of A4
    N = size(sn,2);
    p1 = 0;
    p2 = 0;
    p3 = 0;
    p4 = 0;
    sn(7,:) = 0;

    for i=1:1:N
        if (sn(9,i) ~= 0)
            if (i == 1)
                switch (sn(5,i))
                    case 1
                        sn(7,i) = 1;
                        p1 = i;
                    case 2
                        sn(7,i) = 1;
                        p2 = i;
                    case 3
                        sn(7,i) = 1;
                        p3 = i;
                    case 4
                        sn(7,i) = 1;
                        p4 = i;
                end
            elseif (sn(3,i) == sn(3,i-1)) && (sn(4,i) == sn(4,i-1))
                switch (sn(5,i))
                    case 1
                        if (p1 == 0)
                            sn(7,i) = 1;
                            p1 = i;
                        elseif(sn(6,p1) < sn(6,i))
                            sn(7,i) = 1; % activate
                            sn(7,p1) = 0; % reset
                            p1 = i; % keeping
                        end
                    case 2
                        if (p2 == 0)
                            sn(7,i) = 1;
                            p2 = i;
                        elseif(sn(6,p2) < sn(6,i))
                            sn(7,i) = 1; % activate
                            sn(7,p2) = 0; % reset
                            p2 = i; % keeping
                        end
                    case 3
                        if (p3 == 0)
                            sn(7,i) = 1;
                            p3 = i;
                        elseif(sn(6,p3) < sn(6,i))
                            sn(7,i) = 1; % activate
                            sn(7,p3) = 0; % reset
                            p3 = i; % keeping
                        end
            end
```
case 4
    if (p4 == 0)
        sn(7,i) = 1;
        p4 = i;
    elseif(sn(6,p4) < sn(6,i))
        sn(7,i) = 1; % activate
        sn(7,p4) = 0; % reset
        p4 = i; % keeping
    end
end
else
    switch (sn(5,i))
        case 1
            sn(7,i) = 1;
            p1 = i;
            p2=0; p3=0; p4=0;
        case 2
            sn(7,i) = 1;
            p2 = i;
            p1=0; p3=0; p4=0;
        case 3
            sn(7,i) = 1;
            p3 = i;
            p1=0; p2=0; p4=0;
        case 4
            sn(7,i) = 1;
            p4 = i;
            p1=0; p2=0; p3=0;
    end
end
end
Appendix C

Programming Code of the 3-Sym Method

This section provides the programming code of the 3-Sym method. This method virtually partitions the network with the hexagon tessellation as same as the 6-Tri method. The programing code of this state has been already expressed in Appendix A and the rest, including the independent sensor selection and the dependent sensor selection, is shown below.

**TABLE C–1: SENSOR SELECTION OF THE 3-SYM METHOD**

```matlab
function [ sn ] = A5_3Sym(sn,wide,length,Rs)

%The Algorithm #5: 3 Pieces Symmetrical area (3-Sym)
% sn(x,y,hx,hy,piece,energy,modeFlag)

N = size(sn,2);
%sn = Eng(sn);   %Energy Consumption for one round
sn(7,:) = 0;
p = 0; % an original piece
n = 0; % temporarily selected node
o = 0; % firt node of any cell
e = 0; % last node of any cell
di = 0; %distance between sensor and centre's hex
theta = 0; % an angle that sensor makes upou hex's centre
ax = 0; ay = 0;
bx = 0; by = 0;
cx = 0; cy = 0;
a = 0; % a is original active node acting as the independent sensor
b = 0; % active node acting as a dependent sensor
c = 0; % active node acting as a dependent sensor
pb = 0; % piece of b
pc = 0; % piece of c

for i=1:1:N
    if(sn(6,i) ~= 0)
        if (o == 0) && (i ~= 1)
            o = i;
            a = i;
        end
        if (i == 1)
            o = i;
            a = i;
        elseif (sn(3,i) == sn(3,o)) && (sn(4,i) == sn(4,o)) % the same cell
            if (sn(6,i) > sn(6,a) && (sn(6,i) ~= 0)) % finding the most
```
energy within the same cell (the independent sensor)
a = i;  %keeping
end
if (sn(3,i) == sn(3,N)) && (sn(4,i) == sn(4,N))
e = N;
%%%%%%% finding to b and c the reaches nodes within the save pice
%%%%%%% within a's cell (the dependent sensors)%%%%%%%
 eb = 0;
ec = 0;
switch (sn(5,a))
case 0
  pb = 2;
  pc = 4;
case 1
  pb = 3;
  pc = 5;
case 2
  pb = 0;
  pc = 4;
case 3
  pb = 1;
  pc = 5;
case 4
  pb = 0;
  pc = 2;
case 5
  pb = 1;
  pc = 3;
end
for j=0:1:e
  if (sn(6,i) ~= 0)
    if (sn(5,j) ~= sn(5,a)) && (mod(sn(5,j),2) == mod(sn(5,a),2)) % not the same piece of a and the same odd or even pieces
      if ((sn(6,j) > eb) && (sn(5,j) == pb))
        b = j; % set this node to be active
        eb = sn(6,j);
      elseif ((sn(6,j) > ec) && (sn(5,j) == pc) )
        c = j; % set this node to be active
        ec = sn(6,j);
      end
    end
  end
end
end

%%%%%% Assigning active nodes a, b and c %%%%%%%%%%%
if (a ~= 0)
  sn(7,a) = 1;
end
if (b ~= 0)
  sn(7,b) = 1;
end
if (c ~= 0)
  sn(7,c) = 1;
end
elseif (sn(3,i) ~= sn(3,o)) || (sn(4,i) ~= sn(4,o) || (i == N))
  % found new cell then we consider the last cell for two sensors
%%% finding di(0,a) and theta(a)%%%%%
if (i == N)
  e = N;
else
  e = i-1;
end

%%% finding to b and c the reaches nodes within the
save pice
%%% within a's cell *******
eb = 0;
ce = 0;

switch (sn(5,a))
  case 0
    pb = 2;
    pc = 4;
  case 1
    pb = 3;
    pc = 5;
  case 2
    pb = 0;
    pc = 4;
  case 3
    pb = 1;
    pc = 5;
  case 4
    pb = 0;
    pc = 2;
  case 5
    pb = 1;
    pc = 3;
end

for j=o:1:e
  if (sn(6,i) ~= 0)
    if (sn(5,j) ~= sn(5,a)) && (mod(sn(5,j),2) == mod(sn(5,a),2))
      % not the same piece of a and the same odd or even pieces
      if ((sn(6,j) > eb) && (sn(5,j) == pb))
        b = j; % set this node to be active
        eb = sn(6,j);
      elseif ((sn(6,j) > ec) && (sn(5,j) == pc))
        c = j; % set this node to be active
        ec = sn(6,j);
      end
    end
  end
end

%%% Assigning active nodes a, b and c %%%%%%%%%%%%
if (a ~= 0)
  sn(7,a) = 1;
end
if (b ~= 0)
  sn(7,b) = 1;
end
if (c ~= 0)
  sn(7,c) = 1;
end

%%%%%%% seting for the next cell %%%%%%%%%
if (i ~= N)
o = i;
a = i;
b = 0;
c = 0;
e = 0;
end
else
end
end
end
Appendix D

Programming Code of the *O-Sym* Method

This section provides the programming code of the *O-Sym* method. This method performs the 3-Sym algorithm to select active sensors, and then optimises the coverage efficiency. Hence, the programming code shown below is only the second part (coverage optimisation).

**TABLE D–1: SENSOR SELECTION OF THE O-SYM METHOD**

```matlab
function [ sn ] = ECA_opt(sn,wide,length,Rs)
    %Optimisation efficient coverage area
    % sn(x,y,hx,hy,piece,energy,modeFlag,required)
    % an(x,y,hx,hy,piece,energy,modeFlag,required,sychonous-sn)
    % P(x,y,maker1,maker2,cover,checker), maker:0 is Vertex,1,2,... is a node

    N = size(sn,2);
    an(7,sum(sn(7,:))) = 0; % assigning an for the size of active nodes
    A = 0; B = 0; C = 0; D = 0; E = 0; F = 0; O=0;
    %vertices of Hex
    CP(1:6,1:6) = 0;
    % covered points for checking coverage on hexagon
    Ne = 3*6; % maximum of neighbours
    numNe = 0;
    %number of neighbour of each node
    st = 0; % status of an considered node, where +1 when found a point and
    -1 when a point is covered
    as = 0; % associated nodes when they cover a part of interest Hex.

    %%% filtering only active nodes into an and finding vertices of Hex in
    %%% each cell
    j = 1;
    for i=1:1:N
        if (sn(7,i) == 1)
            an(1:7,j) = sn(1:7,i); %copy sn to an
            an(9,j) = i; % keeping synchronise with sn
            an(8,j) = 0; %this node is required from other if it is 1
            % finding vertices on hexagon
            A(1,j) = an(3,j) + Rs;
            A(2,j) = an(4,j);
            B(1,j) = an(3,j) + Rs/2;
            B(2,j) = an(4,j) + (sqrt(3)*Rs)/2;
            C(1,j) = an(3,j) - Rs/2;
            C(2,j) = an(4,j) + (sqrt(3)*Rs)/2;
            D(1,j) = an(3,j) - Rs;
            D(2,j) = an(4,j);
            E(1,j) = an(3,j) - Rs/2;
            E(2,j) = an(4,j) - (sqrt(3)*Rs)/2;
            F(1,j) = an(3,j) + Rs/2;
            F(2,j) = an(4,j) - (sqrt(3)*Rs)/2;
            O(1,j) = an(3,j);
            O(2,j) = an(4,j);
        end
    end
```
st(j) = 7; % increase new points A to F and O
j = j + 1;
end
end

%%% Finding neighbours (ne) of each active node i
M = size(an,2);
ne(1:M,1:Ne) = 0; % neighbours of active nodes
for i=1:1:M
  k = 1; % index of neighbour
  if (an(8,i) ~= 1) % it is not a required node from others nor the
    same cell of previous one
    for j=1:1:M % checking distance between node and vertices of Hex
      djA = sqrt((an(1,j)-A(1,i))^2 + (an(2,j)-A(2,i))^2);
      djB = sqrt((an(1,j)-B(1,i))^2 + (an(2,j)-B(2,i))^2);
      djC = sqrt((an(1,j)-C(1,i))^2 + (an(2,j)-C(2,i))^2);
      djD = sqrt((an(1,j)-D(1,i))^2 + (an(2,j)-D(2,i))^2);
      djE = sqrt((an(1,j)-E(1,i))^2 + (an(2,j)-E(2,i))^2);
      djF = sqrt((an(1,j)-F(1,i))^2 + (an(2,j)-F(2,i))^2);
      if ((i ~= j) && (an(5,i) ~= an(5,j))) % % not the same node
        if (djA <= Rs) || (djB <= Rs) || (djC <= Rs) || (djD <= Rs) || (djE <= Rs) || (djF <= Rs) % every node with <Rs distance is accounted to be a neighbour of i
          % finding neighbours
          ne(i,k) = j;
          k = k + 1;
        end
      end
    end
  end
  numNe(i) = k - 1;
end

%%% Finding points intersect with Hex and between neighbours, and finally
%%% these points whether are covered or not
for i=1:1:M
  u = 1; % index of point that node i has
  K = numNe(i);
  if (an(8,i) ~= 1)
    switch (an(5,i))
      case 0
        u = 1;
        if (abs(A(1,i)) <= wide/2) && (abs(A(2,i)) <= length/2)
          P(1,u) = A(1,i);
          P(2,u) = A(2,i);
          P(3:6,u) = 0;
          u = u+1;
        end
        if (abs(B(1,i)) <= wide/2) && (abs(B(2,i)) <= length/2)
Appendix D: Programming Code of the O-Sym Method

\begin{verbatim}

P(1, u) = B(1, i);
P(2, u) = B(2, i);
P(3:6, u) = 0;

u = u+1;
end

if (abs(C(1, i)) <= wide/2) && (abs(C(2, i)) <= length/2)

P(1, u) = C(1, i);
P(2, u) = C(2, i);
P(3:6, u) = 0;

u = u+1;
end

if (abs(F(1, i)) <= wide/2) && (abs(F(2, i)) <= length/2)

P(1, u) = F(1, i);
P(2, u) = F(2, i);
P(3:6, u) = 0;

u = u+1;
end

if (abs(O(1, i)) <= wide/2) && (abs(O(2, i)) <= length/2)

P(1, u) = O(1, i);
P(2, u) = O(2, i);
P(3:6, u) = 0;

u = u+1;
end

v = 1; % index of node covering vertices

%%% Finding points intersect with Hex

for k = 1:K

% finding a neighbour make across point with Hex

cell

% Proving at point A

dkA = sqrt((an(1, ne(i, k)) - A(1, i))^2 +
(an(2, ne(i, k)) - A(2, i))^2);

if (dkA <= Rs)

if (v == 1) % avoid duplicated nodes

as(v) = ne(i, k);
v = v+1;

elseif (as(v-1) ~= ne(i, k))

as(v) = ne(i, k);
v = v+1;

end

[P u] = AB(an, ne, A, B, i, k, u, Rs, P, wide, length);

% finding A-B

% finding A-F

end

% Proving at point B

dkB = sqrt((an(1, ne(i, k)) - B(1, i))^2 +
(an(2, ne(i, k)) - B(2, i))^2);

if (dkB <= Rs)

if (v == 1)

as(v) = ne(i, k);
v = v+1;

elseif (as(v-1) ~= ne(i, k))

as(v) = ne(i, k);

end

end

\end{verbatim}
\begin{verbatim}
v = v+1;
end

% finding B-A
[P u] = BA(an,ne,B,A,i,k,u,Rs,P,wid,\text{length});

% finding B-C
end

% Proving at point C
dkC = \sqrt{(an(1,ne(i,k))-C(1,i))^2 + (an(2,ne(i,k))-C(2,i))^2};
if (dkC \leq Rs)
  if (v == 1)
    as(v) = ne(i,k);
    v = v+1;
  elseif (as(v-1) \neq ne(i,k))
    as(v) = ne(i,k);
    v = v+1;
  end

% finding C-B
[P u] = CB(an,ne,C,B,i,k,u,Rs,P,wid,\text{length});

% finding C-O
end

% Proving at point F
dkF = \sqrt{(an(1,ne(i,k))-F(1,i))^2 + (an(2,ne(i,k))-F(2,i))^2};
if (dkF \leq Rs)
  if (v == 1)
    as(v) = ne(i,k);
    v = v+1;
  elseif (as(v-1) \neq ne(i,k))
    as(v) = ne(i,k);
    v = v+1;
  end

% finding F-A
[P u] = FA(an,ne,F,A,i,k,u,Rs,P,wid,\text{length});

% finding F-O
end

% Proving at point O
dkO = \sqrt{(an(1,ne(i,k))-O(1,i))^2 + (an(2,ne(i,k))-O(2,i))^2};
if (dkO \leq Rs)
  if (v == 1)
    as(v) = ne(i,k);
    v = v+1;
  elseif (as(v-1) \neq ne(i,k))
    as(v) = ne(i,k);
    v = v+1;
  end

% finding O-C
[P u] = OC(an,ne,O,C,i,k,u,Rs,P,wid,\text{length});

% finding O-F
end
end
\end{verbatim}
case 1
  u = 1;
  if (abs(A(1,i)) <= wide/2) && (abs(A(2,i)) <= length/2)
    P(1,u) = A(1,i);
    P(2,u) = A(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(B(1,i)) <= wide/2) && (abs(B(2,i)) <= length/2)
    P(1,u) = B(1,i);
    P(2,u) = B(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(C(1,i)) <= wide/2) && (abs(C(2,i)) <= length/2)
    P(1,u) = C(1,i);
    P(2,u) = C(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(D(1,i)) <= wide/2) && (abs(D(2,i)) <= length/2)
    P(1,u) = D(1,i);
    P(2,u) = D(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(O(1,i)) <= wide/2) && (abs(O(2,i)) <= length/2)
    P(1,u) = O(1,i);
    P(2,u) = O(2,i);
    P(3:6,u) = 0;
    u = u+1;
end

v = 1;

for k=1:1:K
  % at point A
  dkA = sqrt((an(1,ne(i,k))-A(1,i))^2 + (an(2,ne(i,k))-A(2,i))^2);
  if (dkA <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [F u] = AB(an,ne,A,B,i,k,u,Rs,P,wide,length);
    [F u] = AO(an,ne,A,O,i,k,u,Rs,P,wide,length);
  end
  % at point B
  dkB = sqrt((an(1,ne(i,k))-B(1,i))^2 + (an(2,ne(i,k))-B(2,i))^2);
  if (dkB <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
v = v+1;

elseif (as(v-1) ~= ne(i,k))
    as(v) = ne(i,k);
    v = v+1;
end;

[P u] = BA(an,ne,B,A,i,k,u,Rs,P,wide,length);
[P u] = BC(an,ne,B,C,i,k,u,Rs,P,wide,length);

% at point C
dkC = sqrt((an(1,ne(i,k))-C(1,i))^2 +
(an(2,ne(i,k))-C(2,i))^2);

if (dkC <= Rs)
    if (v == 1)
        as(v) = ne(i,k);
        v = v+1;
    elseif (as(v-1) ~= ne(i,k))
        as(v) = ne(i,k);
        v = v+1;
end;

[P u] = CB(an,ne,C,B,i,k,u,Rs,P,wide,length);
[P u] = CD(an,ne,C,D,i,k,u,Rs,P,wide,length);

% at point D
dkD = sqrt((an(1,ne(i,k))-D(1,i))^2 +
(an(2,ne(i,k))-D(2,i))^2);

if (dkD <= Rs)
    if (v == 1)
        as(v) = ne(i,k);
        v = v+1;
    elseif (as(v-1) ~= ne(i,k))
        as(v) = ne(i,k);
        v = v+1;
end;

[P u] = DC(an,ne,D,C,i,k,u,Rs,P,wide,length);
[P u] = DO(an,ne,D,O,i,k,u,Rs,P,wide,length);

% at point O
dkO = sqrt((an(1,ne(i,k))-O(1,i))^2 +
(an(2,ne(i,k))-O(2,i))^2);

if (dkO <= Rs)
    if (v == 1)
        as(v) = ne(i,k);
        v = v+1;
    elseif (as(v-1) ~= ne(i,k))
        as(v) = ne(i,k);
        v = v+1;
end;

[P u] = OA(an,ne,O,A,i,k,u,Rs,P,wide,length);
[P u] = OD(an,ne,O,D,i,k,u,Rs,P,wide,length);

end

case 2
    u = 1;
    if (abs(B(1,i)) <= wide/2) && (abs(B(2,i)) <= length/2)

P(1,u) = B(1,i);
P(2,u) = B(2,i);
P(3:6,u) = 0;
u = u+1;
end

if (abs(C(1,i)) <= wide/2) && (abs(C(2,i)) <= length/2}
Appendix D: Programming Code of the O-Sym Method

\[
P(1, u) = C(1, i); \\
P(2, u) = C(2, i); \\
P(3:6, u) = 0; \\
u = u + 1;
\]

\[
\text{if (abs(D(1, i)) <= wide/2) && (abs(D(2, i)) <= length/2)}
\]
\[
P(1, u) = D(1, i); \\
P(2, u) = D(2, i); \\
P(3:6, u) = 0; \\
u = u + 1;
\]

\[
\text{if (abs(E(1, i)) <= wide/2) && (abs(E(2, i)) <= length/2)}
\]
\[
P(1, u) = E(1, i); \\
P(2, u) = E(2, i); \\
P(3:6, u) = 0; \\
u = u + 1;
\]

\[
\text{if (abs(O(1, i)) <= wide/2) && (abs(O(2, i)) <= length/2)}
\]
\[
P(1, u) = O(1, i); \\
P(2, u) = O(2, i); \\
P(3:6, u) = 0; \\
u = u + 1;
\]

\[v = 1;\]

\[\text{for } k=1:1:K\]
\[\% \text{at point } B\]
\[dkB = \sqrt{(an(1, ne(i, k)) - B(1, i))^2 + (an(2, ne(i, k)) - B(2, i))^2};\]
\[\text{if (dkB <= Rs)}\]
\[\quad \text{if (v == 1)}\]
\[\quad \quad \text{as(v) = ne(i, k)}; \]
\[\quad \quad v = v + 1;\]
\[\quad \text{elseif (as(v-1) ~= ne(i, k))}\]
\[\quad \quad \text{as(v) = ne(i, k)}; \]
\[\quad \quad v = v + 1;\]
\[\quad \text{end}\]
\[\quad [P u] = BC(an, ne, B, C, i, k, u, Rs, P, wide, length);\]
\[\quad [P u] = BO(an, ne, B, O, i, k, u, Rs, P, wide, length);\]
\[\text{end}\]
\[\text{\% at point } C\]
\[dkC = \sqrt{(an(1, ne(i, k)) - C(1, i))^2 + (an(2, ne(i, k)) - C(2, i))^2};\]
\[\text{if (dkC <= Rs)}\]
\[\quad \text{if (v == 1)}\]
\[\quad \quad \text{as(v) = ne(i, k)}; \]
\[\quad \quad v = v + 1;\]
\[\quad \text{elseif (as(v-1) ~= ne(i, k))}\]
\[\quad \quad \text{as(v) = ne(i, k)}; \]
\[\quad \quad v = v + 1;\]
\[\quad \text{end}\]
\[\quad [P u] = CB(an, ne, C, B, i, k, u, Rs, P, wide, length);\]
\[\quad [P u] = CD(an, ne, C, D, i, k, u, Rs, P, wide, length);\]
\[\text{end}\]
\[\text{\% at point } D\]
\[dkD = \sqrt{(an(1, ne(i, k)) - D(1, i))^2 + (an(2, ne(i, k)) - D(2, i))^2};\]
Appendix D: Programming Code of the O-Sym Method

\[
\text{if } (dkD \leq Rs) \\
\quad \text{if } (v == 1) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \text{elseif } (as(v-1) \neq ne(i,k)) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \end{if}
\]

\[
\[P_u\] = DC(an,ne,D,C,i,k,u,Rs,P,wide,length); \\
\[P_u\] = DE(an,ne,D,E,i,k,u,Rs,P,wide,length);
\]

\[
% \text{ at point E} \\
\text{dkE} = \sqrt{(an(1,ne(i,k))-E(1,i))^2 + (an(2,ne(i,k))-E(2,i))^2};
\]

\[
\text{if } (dkE \leq Rs) \\
\quad \text{if } (v == 1) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \text{elseif } (as(v-1) \neq ne(i,k)) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \end{if}
\]

\[
\[P_u\] = ED(an,ne,E,D,i,k,u,Rs,P,wide,length); \\
\[P_u\] = EO(an,ne,E,O,i,k,u,Rs,P,wide,length);
\]

\[
% \text{ at point O} \\
\text{dkO} = \sqrt{(an(1,ne(i,k))-O(1,i))^2 + (an(2,ne(i,k))-O(2,i))^2};
\]

\[
\text{if } (dkO \leq Rs) \\
\quad \text{if } (v == 1) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \text{elseif } (as(v-1) \neq ne(i,k)) \\
\quad \quad \text{as}(v) = ne(i,k); \\
\quad \quad v = v+1; \\
\quad \end{if}
\]

\[
\[P_u\] = OB(an,ne,O,B,i,k,u,Rs,P,wide,length); \\
\[P_u\] = OE(an,ne,O,E,i,k,u,Rs,P,wide,length);
\]

\[
\text{case 3} \\
\text{u} = 1; \\
\text{if } (abs(C(1,i)) \leq \text{width}/2) \&\& (abs(C(2,i)) \leq \text{length}/2)
\]

\[
P(1,u) = C(1,i); \\
P(2,u) = C(2,i); \\
P(3:6,u) = 0; \\
u = u+1;
\]

\[
\text{if } (abs(D(1,i)) \leq \text{width}/2) \&\& (abs(D(2,i)) \leq \text{length}/2)
\]

\[
P(1,u) = D(1,i); \\
P(2,u) = D(2,i); \\
P(3:6,u) = 0; \\
u = u+1;
\]

\[
\text{if } (abs(E(1,i)) \leq \text{width}/2) \&\& (abs(E(2,i)) \leq \text{length}/2)
\]

\[
P(1,u) = E(1,i); \\
P(2,u) = E(2,i);
\]
\[ P(3:6, u) = 0; \]
\[ u = u + 1; \]

\textbf{end}

\textbf{if} (abs(F(1,i)) <= wide/2) && (abs(F(2,i)) <= length/2 )

\[ P(1, u) = F(1, i); \]
\[ P(2, u) = F(2, i); \]
\[ P(3:6, u) = 0; \]
\[ u = u + 1; \]

\textbf{end}

\textbf{if} (abs(O(1,i)) <= wide/2) && (abs(O(2,i)) <= length/2 )

\[ P(1, u) = O(1, i); \]
\[ P(2, u) = O(2, i); \]
\[ P(3:6, u) = 0; \]
\[ u = u + 1; \]

\textbf{end}

\[ v = 1; \]

\textbf{for} k=1:1:K

\% at point C

\[ dkC = \sqrt{(an(1,ne(i,k)) - C(1,i))^2 + (an(2,ne(i,k)) - C(2,i))^2}; \]

\textbf{if} (dkC <= Rs)

\textbf{if} (v == 1)

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{elseif} (as(v-1) == ne(i,k))

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{end}

\[ [P u] = CO(an, ne, C, O, i, k, u, Rs, P, wide, length); \]
\[ [P u] = CD(an, ne, C, D, i, k, u, Rs, P, wide, length); \]

\textbf{end}

\% at point D

\[ dkD = \sqrt{(an(1,ne(i,k)) - D(1,i))^2 + (an(2,ne(i,k)) - D(2,i))^2}; \]

\textbf{if} (dkD <= Rs)

\textbf{if} (v == 1)

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{elseif} (as(v-1) == ne(i,k))

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{end}

\[ [P u] = DC(an, ne, D, C, i, k, u, Rs, P, wide, length); \]
\[ [P u] = DE(an, ne, D, E, i, k, u, Rs, P, wide, length); \]

\textbf{end}

\% at point E

\[ dkE = \sqrt{(an(1,ne(i,k)) - E(1,i))^2 + (an(2,ne(i,k)) - E(2,i))^2}; \]

\textbf{if} (dkE <= Rs)

\textbf{if} (v == 1)

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{elseif} (as(v-1) == ne(i,k))

\[ as(v) = ne(i,k); \]
\[ v = v + 1; \]

\textbf{end}

\[ [P u] = ED(an, ne, E, D, i, k, u, Rs, P, wide, length); \]
[P u] = EF(an,ne,E,F,i,k,u,Rs,P,wide,length);
end

% at point F
  dkF = sqrt((an(1,ne(i,k))-F(1,i))^2 +
  (an(2,ne(i,k))-F(2,i))^2);
  if (dkF <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = FE(an,ne,E,F,i,k,u,Rs,P,wide,length);
    [P u] = FO(an,ne,F,O,i,k,u,Rs,P,wide,length);
  end

% at point O
  dkO = sqrt((an(1,ne(i,k))-O(1,i))^2 +
  (an(2,ne(i,k))-O(2,i))^2);
  if (dkO <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = OC(an,ne,O,C,i,k,u,Rs,P,wide,length);
    [P u] = OF(an,ne,O,F,i,k,u,Rs,P,wide,length);
  end
end

case 4
  u = 1;
  if (abs(A(1,i)) <= wide/2) && (abs(A(2,i)) <= length/2)
    P(1,u) = A(1,i);
    P(2,u) = A(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(D(1,i)) <= wide/2) && (abs(D(2,i)) <= length/2)
    P(1,u) = D(1,i);
    P(2,u) = D(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(E(1,i)) <= wide/2) && (abs(E(2,i)) <= length/2)
    P(1,u) = E(1,i);
    P(2,u) = E(2,i);
    P(3:6,u) = 0;
    u = u+1;
  end
  if (abs(F(1,i)) <= wide/2) && (abs(F(2,i)) <= length/2)
    P(1,u) = F(1,i);
    P(2,u) = F(2,i);
    P(3:6,u) = 0;
    u = u+1;
end
if (abs(O(1,i)) <= wide/2 && abs(O(2,i)) <= length/2)
    P(1,u) = O(1,i);
P(2,u) = O(2,i);
P(3:6,u) = 0;
u = u+1;
end

v = 1;

for k=1:1:K
    % at point A
    dkA = sqrt((an(1,ne(i,k))-A(1,i))^2 + (an(2,ne(i,k))-A(2,i))^2);
    if (dkA <= Rs)
        if (v == 1)
            as(v) = ne(i,k);
v = v+1;
        elseif (as(v-1) ~= ne(i,k))
            as(v) = ne(i,k);
v = v+1;
        end
        [P u] = AO(an,ne,A,O,i,k,u,Rs,P,wide,length);
        [P u] = AF(an,ne,A,F,i,k,u,Rs,P,wide,length);
    end
    % at point D
    dkD = sqrt((an(1,ne(i,k))-D(1,i))^2 + (an(2,ne(i,k))-D(2,i))^2);
    if (dkD <= Rs)
        if (v == 1)
            as(v) = ne(i,k);
v = v+1;
        elseif (as(v-1) ~= ne(i,k))
            as(v) = ne(i,k);
v = v+1;
        end
        [P u] = DE(an,ne,D,E,i,k,u,Rs,P,wide,length);
        [P u] = DO(an,ne,D,O,i,k,u,Rs,P,wide,length);
    end
    % at point E
    dkE = sqrt((an(1,ne(i,k))-E(1,i))^2 + (an(2,ne(i,k))-E(2,i))^2);
    if (dkE <= Rs)
        if (v == 1)
            as(v) = ne(i,k);
v = v+1;
        elseif (as(v-1) ~= ne(i,k))
            as(v) = ne(i,k);
v = v+1;
        end
        [P u] = ED(an,ne,E,D,i,k,u,Rs,P,wide,length);
        [P u] = EF(an,ne,E,F,i,k,u,Rs,P,wide,length);
    end
    % at point F
    dkF = sqrt((an(1,ne(i,k))-F(1,i))^2 + (an(2,ne(i,k))-F(2,i))^2);
    if (dkF <= Rs)
        if (v == 1)
            as(v) = ne(i,k);
v = v+1;
        elseif (as(v-1) ~= ne(i,k))
            as(v) = ne(i,k);
v = v+1;
        end
        [P u] = FD(an,ne,F,D,i,k,u,Rs,P,wide,length);
        [P u] = EF(an,ne,F,E,i,k,u,Rs,P,wide,length);
    end
end
Appendix D: Programming Code of the O-Sym Method

\[
\begin{align*}
\text{as}(v) &= \text{ne}(i, k); \\
v &= v+1; \\
\end{align*}
\]

end

\[
\begin{align*}
[P]\ &= \text{FA}(\text{an}, \text{ne}, F, A, i, k, u, Rs, P, \text{wide}, \text{length}); \\
[P]\ &= \text{FE}(\text{an}, \text{ne}, F, E, i, k, u, Rs, P, \text{wide}, \text{length}); \\
\end{align*}
\]

end

% at point O
\[
(\text{an}(2, \text{ne}(i, k)) - O(2, i))^2 +
\]

\[
\begin{align*}
(\text{an}(1, \text{ne}(i, k)) - O(1, i))^2 &> 0; \\
\text{if} & (dKO \leq Rs) \\
\text{if} & (v == 1) \\
\text{as}(v) &= \text{ne}(i, k); \\
v &= v+1; \\
\text{elseif} & (\text{as}(v-1) \neq \text{ne}(i, k)) \\
\text{as}(v) &= \text{ne}(i, k); \\
v &= v+1; \\
\end{align*}
\]

\[
\begin{align*}
[P]\ &= \text{OA}(\text{an}, \text{ne}, O, A, i, k, u, Rs, P, \text{wide}, \text{length}); \\
[P]\ &= \text{OD}(\text{an}, \text{ne}, O, D, i, k, u, Rs, P, \text{wide}, \text{length}); \\
\end{align*}
\]

case 5
\[
\begin{align*}
u &= 1; \\
f & (\text{abs}(A(1, i)) \leq \text{wide}/2) \& \& (\text{abs}(A(2, i)) \leq \text{length}/2) \\
P(1, u) &= A(1, i); \\
P(2, u) &= A(2, i); \\
P(3:6, u) &= 0; \\
u &= u+1; \\
\end{align*}
\]

\[
\begin{align*}
f & (\text{abs}(B(1, i)) \leq \text{wide}/2) \& \& (\text{abs}(B(2, i)) \leq \text{length}/2) \\
P(1, u) &= B(1, i); \\
P(2, u) &= B(2, i); \\
P(3:6, u) &= 0; \\
u &= u+1; \\
\end{align*}
\]

\[
\begin{align*}
f & (\text{abs}(E(1, i)) \leq \text{wide}/2) \& \& (\text{abs}(E(2, i)) \leq \text{length}/2) \\
P(1, u) &= E(1, i); \\
P(2, u) &= E(2, i); \\
P(3:6, u) &= 0; \\
u &= u+1; \\
\end{align*}
\]

\[
\begin{align*}
f & (\text{abs}(F(1, i)) \leq \text{wide}/2) \& \& (\text{abs}(F(2, i)) \leq \text{length}/2) \\
P(1, u) &= F(1, i); \\
P(2, u) &= F(2, i); \\
P(3:6, u) &= 0; \\
u &= u+1; \\
\end{align*}
\]

\[
\begin{align*}
f & (\text{abs}(O(1, i)) \leq \text{wide}/2) \& \& (\text{abs}(O(2, i)) \leq \text{length}/2) \\
P(1, u) &= O(1, i); \\
P(2, u) &= O(2, i); \\
P(3:6, u) &= 0; \\
u &= u+1; \\
\end{align*}
\]

v = 1;
for k=1:1:K
  % at point A
  dkA = sqrt((an(1,ne(i,k))-A(1,i))^2 + (an(2,ne(i,k))-A(2,i))^2);
  if (dkA <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = AB(an,ne,A,B,i,k,u,Rs,P,wide,length);
    [P u] = AF(an,ne,A,F,i,k,u,Rs,P,wide,length);
  end
  % at point B
  dkB = sqrt((an(1,ne(i,k))-B(1,i))^2 + (an(2,ne(i,k))-B(2,i))^2);
  if (dkB <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = BA(an,ne,B,A,i,k,u,Rs,P,wide,length);
    [P u] = BO(an,ne,B,O,i,k,u,Rs,P,wide,length);
  end
  % at point E
  dkE = sqrt((an(1,ne(i,k))-E(1,i))^2 + (an(2,ne(i,k))-E(2,i))^2);
  if (dkE <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = EF(an,ne,E,F,i,k,u,Rs,P,wide,length);
    [P u] = EO(an,ne,E,O,i,k,u,Rs,P,wide,length);
  end
  % at point F
  dkF = sqrt((an(1,ne(i,k))-F(1,i))^2 + (an(2,ne(i,k))-F(2,i))^2);
  if (dkF <= Rs)
    if (v == 1)
      as(v) = ne(i,k);
      v = v+1;
    elseif (as(v-1) ~= ne(i,k))
      as(v) = ne(i,k);
      v = v+1;
    end
    [P u] = FA(an,ne,F,A,i,k,u,Rs,P,wide,length);
    [P u] = FE(an,ne,F,E,i,k,u,Rs,P,wide,length);
  end
  % at point O
  dkO = sqrt((an(1,ne(i,k))-O(1,i))^2 + (an(2,ne(i,k))-O(2,i))^2);
Appendix D: Programming Code of the O-Sym Method

```matlab
if (dkO <= Rs)
    if (v == 1)
        as(v) = ne(i,k);
        v = v+1;
    elseif (as(v-1) ~= ne(i,k))
        as(v) = ne(i,k);
        v = v+1;
    end
    [P u] = OB(an,ne,O,B,i,k,u,Rs,P,wide,length);
    [P u] = OE(an,ne,O,E,i,k,u,Rs,P,wide,length);
end
end
end

%%%%%%% Finding intersection points between neighbours
a = 0; % a is a half of duv
K = size(as,2);
L = size(as,2);
for k=1:1:K-1
    for l=k+1:1:L
        dx = abs(an(1,as(k))-an(1,as(l)));  
        dy = abs(an(2,as(k))-an(2,as(l)));
        dkl = sqrt((dx)^2 + (dy)^2);  % distance between nodes k and l
        if (dkl <= 2*Rs) && (dkl ~= 0)
            a = dkl/2;
            h = sqrt(Rs^2 - a^2);
            if (an(1,as(k)) <= an(1,as(l))) && (an(2,as(k)) <= an(2,as(l)))
                xc = an(1,as(k)) + (a*dx/dkl);  %finding (xc,yc)
                yc = an(2,as(k)) + (a*dy/dkl);
                rx = h*dy/dkl;
                ry = h*dx/dkl;
                x1 = xc - rx;
                x2 = xc + rx;
                y1 = yc + ry;
                y2 = yc - ry;
            elseif (an(1,as(k)) <= an(1,as(l))) && (an(2,as(k)) > an(2,as(l))
                xc = an(1,as(k)) + (a*dx/dkl);  %finding (xc,yc)
                yc = an(2,as(k)) - (a*dy/dkl);
                rx = h*dy/dkl;
                ry = h*dx/dkl;
                x1 = xc + rx;
                x2 = xc - rx;
                y1 = yc + ry;
                y2 = yc - ry;
            elseif (an(1,as(k)) > an(1,as(l))) && (an(2,as(k)) <= an(2,as(l))
                xc = an(1,as(k)) - (a*dx/dkl);  %finding (xc,yc)
                yc = an(2,as(k)) - (a*dy/dkl);
                rx = h*dy/dkl;
                ry = h*dx/dkl;
                x1 = xc - rx;
                x2 = xc + rx;
                y1 = yc + ry;
                y2 = yc + ry;
            elseif (an(1,as(k)) > an(1,as(l))) && (an(2,as(k)) > an(2,as(l))
                xc = an(1,as(k)) - (a*dx/dkl);  %finding (xc,yc)
                yc = an(2,as(k)) - (a*dy/dkl);
                rx = h*dy/dkl;
                ry = h*dx/dkl;
                x1 = xc - rx;
                x2 = xc + rx;
                y1 = yc + ry;
                y2 = yc + ry;
            end
        end
    end
end
```

164
<= an(2, as(l)))

xc = an(1, as(k)) - (a * dx/dkl); % finding (xc, yc) of Pc

yc = an(2, as(k)) + (a * dy/dkl);
rx = h * dy/dkl;
ry = h * dx/dkl;
x1 = xc + rx;
x2 = xc - rx;
y1 = yc + ry;
y2 = yc - ry;

end

%%% Filtering only points within Hex’s area

d1H = sqrt((x1 - an(3,i))^2 + (y1 - an(4,i))^2);
d2H = sqrt((x2 - an(3,i))^2 + (y2 - an(4,i))^2);
if (d1H <= Rs) && (abs(x1) <= wide/2) && (abs(y1) <= length/2)
P(1,u) = x1;
P(2,u) = y1;
P(3,u) = as(k);
P(4,u) = as(l);
u = u + 1;
end
if (d2H <= Rs) && (abs(x2) <= wide/2) && (abs(y2) <= length/2)
P(1,u) = x2;
P(2,u) = y2;
P(3,u) = as(k);
P(4,u) = as(l);
u = u + 1;
end

%%% Finding each point whether is covered or not
K = size(P,2);
size(P,1);
L = numNe(i);
for k = 1:1:K
for l = 1:1:L
if (P(3,k) == 0)
dkl = sqrt((P(1,k) - an(1,ne(i,l)))^2 + (P(2,k) - an(2,ne(i,l)))^2);
if (dkl <= Rs)
P(5,k) = ne(i,l); % this node covered by neighbour l
P(6,k) = 1; % this node is covered
break;
end
elseif (P(3,k) ~= 0) && (P(4,k) == 0)
if (P(3,k) ~= ne(i,l))
dkl = sqrt((P(1,k) - an(1,ne(i,l)))^2 + (P(2,k) - an(2,ne(i,l)))^2);
if (dkl <= Rs)
P(5,k) = ne(i,l); % this node covered by neighbour l
P(6,k) = 1; % this node is covered
break;
end
end
end

end

end

end

%%% Finding each point whether is covered or not
break;
end
end
elseif (P(3,k) ~= 0) && (P(4,k) ~= 0)
    if (P(3,k) ~= ne(i,l)) && (P(4,k) ~= ne(i,l))
        dkl = sqrt((P(1,k) - an(1,ne(i,l)))^2 + (P(2,k) - an(2,ne(i,l))))^2);
        if (dkl <= Rs)
            P(5,k) = ne(i,l); % this node covered by neighbour l
            P(6,k) = 1; % this node is covered
            break;
        end
    end
end
end

%%%%% Reseting active nodes and informing those are required
full = size(P,2);
K = size(P,2);
Covered = sum(P(6,:));
if (full == Covered)
    an(7,i) = 0; % reset the node to sleep at an
    sn(7,an(9,i)) = 0; % reset the node to sleep at sn
    for k=1:1:K
        if (an(8,P(5,k)) == 0)
            an(8,P(5,k)) = 1; %set required node
            % sn(8,an(9,P(5,k))) = 1; %set required node
        end
    end
    end
end
P = 0; % reset points
end

end
numAN = size(an,2);
umRD = numAN - sum(an(7,:)); %number of reduced nodes
PerRD = numRD*numAN/100;
References


References


References

2008.


References


References


References


References

51, 2012.


