ISLAMIC UNIVERSITY OF TECHNOLOGY

UNDER GRADUATE THESIS

Cellular Automaton Based Motion Planning for Mobile Wireless Sensor Networks

---

Authors:

Ahnaf Munir (114401)

Shihabuzzaman (114402)

Supervisor:

Dr. Muhammad Mahbub Alam

Professor

Department of Computer Science and Engineering

A thesis submitted to the Department of CSE in fulfilment of the requirements for the Degree of B.Sc Engineering in CSE.

Academic Year: 2014-15.

October, 2015

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigation carried out by Ahnaf Munir and Shihabuzzaman under the supervision of Dr. Muhammad Mahbub Alam in the Department of Computer Science and Engineering (CSE), IUT, Dhaka, Bangladesh. It is also declared that neither of this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Ahnaf Munir
Student ID - 114401

Shihabuzzaman
Student ID - 114402

Supervisor:

Dr. Muhammad Mahbub Alam
Associate Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)

ISLAMIC UNIVERSITY OF TECHNOLOGY

Abstract

B.Sc in Computer Science and Engineering

Cellular Automaton Based Motion Planning for Mobile Wireless Sensor Networks

by Ahnaf Munir (114401)
Shihabuzzaman (114402)

Mobile Wireless Sensor Networks (MWSN) is an ad-hoc network that comprises of a large number of sensors. The sensors usually have limited sensing and communication capabilities. Recent times has seen a rapid development in MWSN technology which has resulted in its increasing application and transformed this sector into an important field of research. Mobility is a key factor in the implementation of MWSN. After being deployed in the environment the sensors need to move around to increase the coverage of the network while maintaining connectivity. This movement is done by the individual sensors based on local information. The use of local information means that Cellular Automaton (CA) is suitable for developing motion planning algorithms for MWSN. CA is a biologically inspired discrete model. Although there are works that have used CA for developing motion planning algorithm for MWSN they mostly used square grids which is not a good representation of the actual MWSN model. In this paper we develop a set of probabilistic and deterministic cellular automaton (CA)-based algorithms for motion planning problems in MWSNs for hexagonal grid. We consider scenarios where sensors are either explicitly or randomly placed in the environment and they need to disperse to maximize the covered area and also maintain connectivity. We conduct simulations for both the deterministic and probabilistic models and find that the probabilistic model yield better results....

Acknowledgements

In full gratitude, we would like to acknowledge the following individuals who encouraged, inspired, supported, assisted, and sacrificed themselves to help our pursuit of the successful thesis work.

From our academy, we would like to thank Dr. Muhammad Mahbub Alam, Professor, CSE, IUT for the continuous support of our thesis work and related research, for his patience, motivation, and immense knowledge. His guidance helped us in all the time of research and writing of this thesis.

We would also like to thanks Md. Sakhawat Hossen, Assistant Professor, CSE, IUT for his insightful comments and encouragement, but also for the hard question which incented us to widen our research from various perspectives.

Last but not the least, we would like to thank our families for supporting us spiritually throughout the writing of this thesis and our life in general.

With Regards,
Ahnaf Munir (114401)
Shihabuzzaman (114402)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Advances in electronic components combined with the need of obtaining information about systems that need to be monitored have allowed the development of new kinds of computer networks. A Mobile Wireless Sensor Network (MWSN) is a special kind of network with distributed sensing and processing capabilities. It can be used in a wide range of applications, such as environmental monitoring, industrial applications, smart transport systems, security of our daily life, environmental monitoring etc. [1, 2, 3]

A MWSN is different from any traditional network. A typical MWSN consists of several tiny mobile nodes which have limited attributes (e.g. energy, transmission range, memory, processing power and so on.). One of the important optimization problems in an MWSN is to maximize the coverage of the network while maintaining coverage. The sensors in a MWSN have movement capabilities. They are deployed in a certain area and they need to move around to increase the coverage of the network while maintaining connectivity among themselves.

The movement of a sensor requires significant amount of energy. Since the sensors have limited power this movement should be restricted as much as possible in order to save energy for sensing and communication. So a balance must be maintained among coverage, connectivity and movement. This problem, called the Mobile Dispersion Problem is what we address here.

The sensors in a MWSN only use local information due to their limited sensing and communication range. The use of small amount of local information makes Cellular Automata (CA) suitable for developing MWSN algorithms and testing them [4-8, 13-15]. CA is biologically inspired model that uses information gathered from its neighbors to make decisions.

1.2 Thesis Contribution

The contribution of this thesis is as follows:

1. All the mobile dispersion algorithms that uses CA were developed for square grid. But square grid is not a good representation of the the actual sensors as compared to hexagonal grid. We use an dispersion algorithm proposed by Salim et. Al [12] for developing an CA-based algorithm for a hexagonal grid. The hexagonal grid gives much better representation of an actual MWSN model.

2. Our initially proposed algorithm contains rules that were developed for a square grid. These rules create certain problems when applied to sensors in a hexagonal grid. So we modify the existing rules to fit a hexagonal grid better.

3. Even after modification of the rules some problems exists in the deterministic model which cause disconnection among the sensors. Such problems are addressed through the use of a probabilistic model instead of the initial deterministic model.

4. The limited energy of the sensors means that their movement should be limited in order to preserve them for other uses such as sensing the environment and communicating with other senors. We develop rules to limit the movement of the sensors which eventually results in preservation of energy.

1.3 Thesis Outline

The paper is organized as follows: Chapter 2 provides a short overview of MWSN and CA; Chapter 3 discusses the state of the art in the field; Chapter 4 discusses our proposed algorithms; Chapter 5 shows the results of our algorithms ; and Chapter 6 concludes the paper and presents future work.

Chapter 2

Background

2.1 Mobile Airless Sensor Network (MWSN)

Mobile Wireless Mobile Networks (MWSNs) are distributed adhoc networks of nodes that can sense, actuate, compute and communicate with each other using point-to-point, multi-hop communication. This kind of network may consist of hundreds to thousands of sensor nodes that have the capability of sensing, processing and communicating using a wireless medium. Nodes in such networks include mobile sensors, robots, and even humans. Such systems combine the most advanced concepts in perception, communication and control to create computational systems capable of large-scale interaction with the environment, extending the individual capabilities of each network component to encompass a much wider area and range of data.

Due to the absence of any networking infrastructure the nodes must cooperate to accomplish communication, global control and distributed information aggregation. MWSNs aims to collect data and sometimes control an environment.

The establishment of a mobile wireless sensor network is illustrated in Figure 2.1. Initially, the sensor nodes are deployed over an area of interest as illustrated in Figures 2.1(a) and 2.1(b). The node deployment can be done, for example, by dropping a large number of sensor nodes from an airplane in a certain area, or placing them in this area by hand or using a robot. They are able to discover their locations (see Figure 2.1 (c)) and organize themselves as a wireless network (see Figure 2.1 (d)). A MWSN must be able to operate under very dynamic conditions.

Once the network is formed and the sensor nodes are operating, most sensor nodes will be able to sustain a steady state of operation, where energy reservoirs will be nearly full, and they will be able to support all the sensing, processing, and communication tasks required. In this mode, sensor nodes will constitute a

multi-hop network. The sensor nodes begin to establish routes by which information passed to one or more sink nodes. Sink nodes are typical sensor nodes that usually differ from other types of sensor nodes in the following aspects: they have more energy, longer radio range and do not perform sensing. When there are structural differences among the nodes we say that the network is heterogeneous. In this work, we just address the homogeneous networks (that is, we consider that all nodes have the same structure and properties).

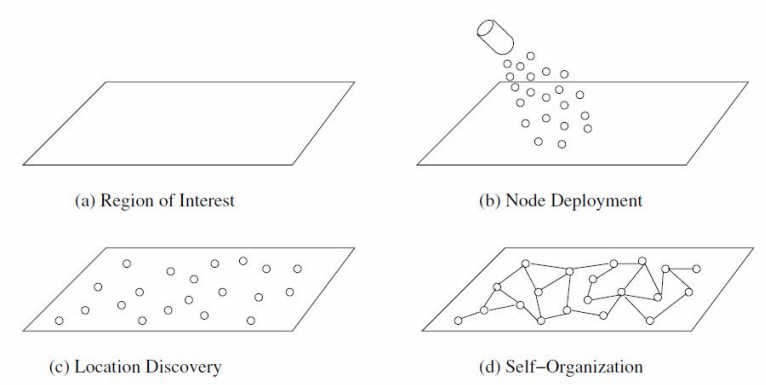

FIGURE 2.1: Creation of a MWSN

2.2 Cellular Automata (CA)

A Cellular Automaton (CA) is a discrete time decentralized system that can be used to model physical systems. It is a biologically inspired model that uses small amount of local information for making decisions.

2.2.1 Formal Definition

A CA can be formally defined as a 4-tuple (L, S, N, δ) where:

- L is a regular grid. The elements that compose this grid are called cells.
- S is a finite set of states
- N is a finite set (of size $|N| = n$) of neighborhood index, such that $\forall c \in L : r + c \in L$
- $\delta : S^n \rightarrow S$ is a transition function

The four components are discussed briefly below:

Grid

The cells in the cellular automata reside on grids. This grid can be of various types. They can be in any finite number of dimensions. The shape can me rectangular, triangular, hexagonal etc. The grid can have either finite or infinite size.

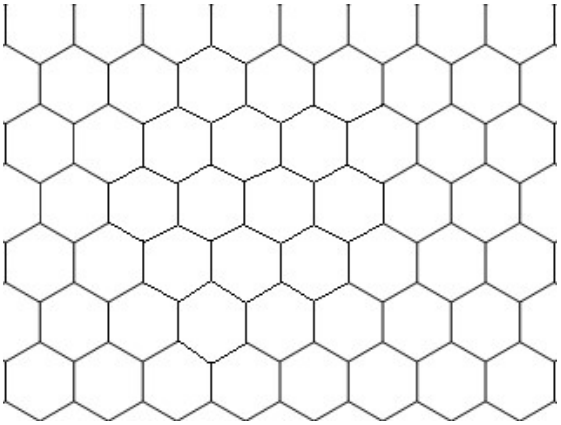

FIGURE 2.2: Hexagonal Grid

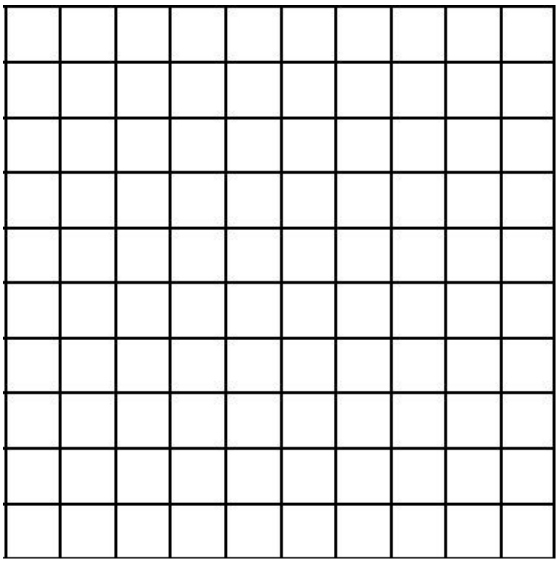

FIGURE 2.3: Hexagonal Grid

Generally, a grid is called regular if it is a coverage of d-dimensional space, that is, the elements of the grid (cells) together complete the d-dimensional space entirely, and the translation of the grid in d independent directions results in the grid itself.

State

A finite set of states is associated with any cellular automata. The cells can be in any of these states. The simplest set of states are the on and off states or 0 and 1. In every time step each of the cell changes or retains its state by consulting the rules that have been defined for that particular automata. The rules usually consider the state of the cell and also of those that are its neighbors (discussed below) in the previous time step.

Transition Function

The trasition function is responsible for changing the states of the cells. The state of a cell c ϵ C, at time $t + 1$, is determined via the transition function δ depending on the current state of c and the states of cells in the neighborhood of c at time t.

Neighbourhood

The neighbors of a cell are the cells that are considered to be adjacent to it. The number of adjacent cells that are considered neighbor depends on that particular cellular automata. In case of a square grid if all the adjacent cells (maximum 8) of a cell are considered as the neighbors, then this is called a Moore neighborhood [9].

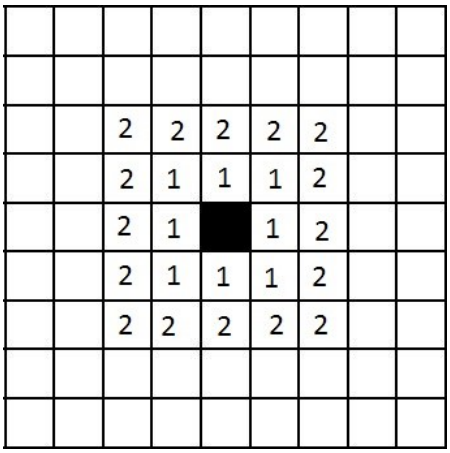

FIGURE 2.4: Neighbors of the central cell in square grid

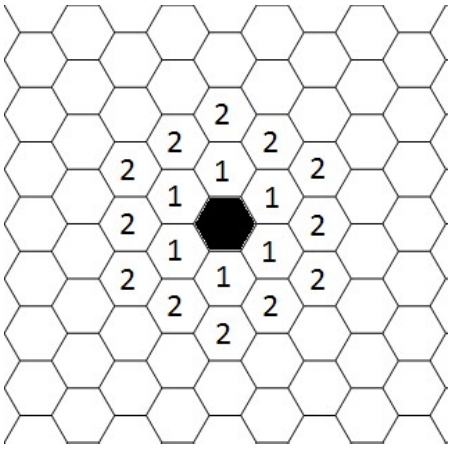

FIGURE 2.5: Neighbors of the central cell in hexagonal grid

In figures 2.4 and 2.5 the neighbors of the central black cell are shown. The number within the cells denote the distance of the neighbors from the central cell

2.3 Range of a Sensor

Each sensor in a MWSN has two kind of range associated with it; the communication range and the sensing range.

The communication range of a sensor is the area surrounding the sensor within which the sensor can communicate with other sensors. So if any sensor is located within the communication range of a sensor then the two sensors can communicate with each other. We represent communication range by R_c.

The communication range of a sensor is the area surrounding the sensor within which it can sense the environment. So a sensor can sense the environmental area that falls within its sensing range. We represent communication range by R_s.

Chapter 3

Literature Review

Our work is based on the CA model that was introduced by Cunha et. Al [7]. Cunha also discussed discussed the applicability of CA for MWSN model and also introduces an algorithm to solve the network topology problem.

The paper by Salim et. Al [6] provides improved algorithm for the problem defined in Cunha. This work greatly increases the lifespan of the network.

Most of the solutions that have been proposed for our problem are either global or distributed algorithms that are also, arguably, highly complex. A simple approach has been proposed in [11]. This paper deals with sensor node placement algorithm running on cellular automaton. The goal of this paper is to achieve adequate coverage, connectivity and sparsity while being resilient to changing environment condition.

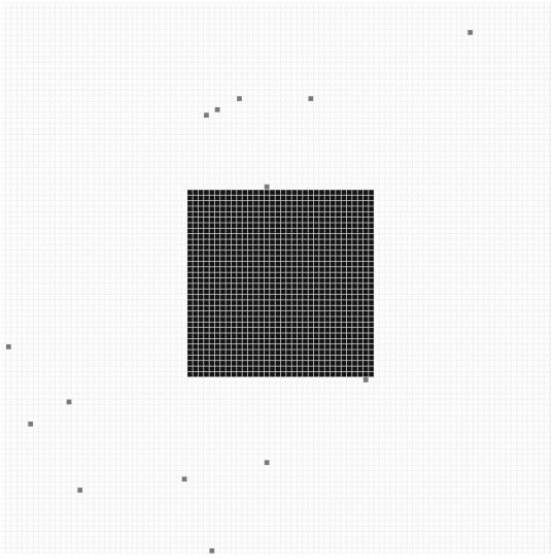

FIGURE 3.1: Initial state of an automaton with 10,000 cells and 729 sensors

Sensors are placed randomly on a square grid and move in two steps. During even cycle each node calculates the weighted distance of every other sensors in the neighborhood. Then the decision to move is taken probabilistically based on weighted distance. The equation used to calculate weighted distance of neighborhood is: $k = 8N_1 + 4N_2 + 2N_3 + 1N_4$

Where N_n is the number of nodes within a distance of n cell from the sensor in question. Probability of movement varies for different value of K.

k is then used to determine the probability of movement:

- For $k = 0$ or $k \geq 8$ the node has a 50% chance of moving
- For $5 \leq k \leq 7$ the node has a 20% chance of moving
- For $1 \leq k \leq 4$ the node has a 5% chance of moving

Once it has taken the decision to move, a sensor chooses at random one of its eight immediate neighboring cells while following two conditions:

- The chosen cell must be empty
- The chosen cell must also be outside the reach of all other nodes

During the odd time step the node actually moves as defined in the even time step.

Papers by Salim et. Al [11], [12] discusses algorithms for moving the nodes in the network. The papers considered grid with square cells where each cell can be either empty or contain one or more nodes. The nodes would then either remain in its cell or move to the adjacent cells according to the algorithm. We discuss the algorithm as our initial rules are based on [12].

The algorithm determines the movement direction of a sensor s based on the weighted number of neighbors of s in the positive and in the negative x-direction (respectively, y-direction). The weights assigned to neighbors in case of $R_c = 3$ are as follows: the weights for neighbors at distances 1, 2 and 3 are 4, 2 and 1, respectively. In case of $R_c = 2$, the weights for neighbors at distances 1 and 2 are 2 and 1, respectively.In an ideal case, one sensor can have neighbors at distance 3 away in case of $R_c = 3$ and $R_s = 1$ to maximize the coverage while maintaining connectivity. For this reason we assign the weights to be inversely proportional to the distance.

The state representing an individual sensor is a pair(x,y),x,y ϵ-1,0,1. The state remembers the last move of the sensor. The direction of movement is

stored in the state because, in order to avoid infinite loops, the algorithm preserves the current movement direction. For example, a pair (0,1) means that in the last time step the sensor did not move in the x-direction and moved upwards along the y-direction. The next movement step of a sensor s is determined by the weighted neighborhood of s and the previous movement direction of s that is stored in the pair of integers representing s. When a cell has more than one sensor, each represented by a pair(x,y),x,y ϵ-1,0,1, the algorithm computes the potential movement direction for each of these sensors. (The movement direction depends on (x, y) and, thus, may be different for different sensors in the same cell.)

The algorithms use a parameter (multiplier), $M >= 2$, that serves to encourage the sensor to keep moving in the direction of its previous movement step. We define the movement rule as follows.

Suppose that a sensor s is represented by a pair (s_x, s_y). Suppose that $w1$ is the sum of weights of neighbors of s in the negative x-direction (to the left of s). Suppose that w_2 is the sum of basic weights of neighbors of s in the positive x-direction (to the right of s). The potential movement of s in x-direction depends on the value of s_x, i.e. by remembering the last move, the sensor tries to keep moving in the same direction, unless there is a really good reason to change the direction. So the movement of a sensor along the x-direction depends on its neighbors in the positive and negative x-direction and the current value of s_x. If we use a multiplier M, then the decision of movement to the x-direction is determined by a value x-move(s) defined as

$$if s_x = 0, x - move(s) = w_2 - w_1$$

$$if s_x = -1, x - move(s) = M \times w2 - w1$$

$$if s_x = 1, x - move(s) = w2 - (M \times w1)$$

Now, if x-move(s) = 0 then the sensor does not move in the x-direction. If x-move(s) >= 1, then the sensor moves to the negative x-direction whereas if x-moves(s) <= 1, then the sensor moves to the positive x-direction. Movement in the y-direction is determined analogously. If we do not have multipliers, that is, set $M = 1$, the movement rules defined by the weighted neighborhood of a sensor together with the move back rules can lead to infinite cycles. This is illustrated by the following example (Figure 3.2).

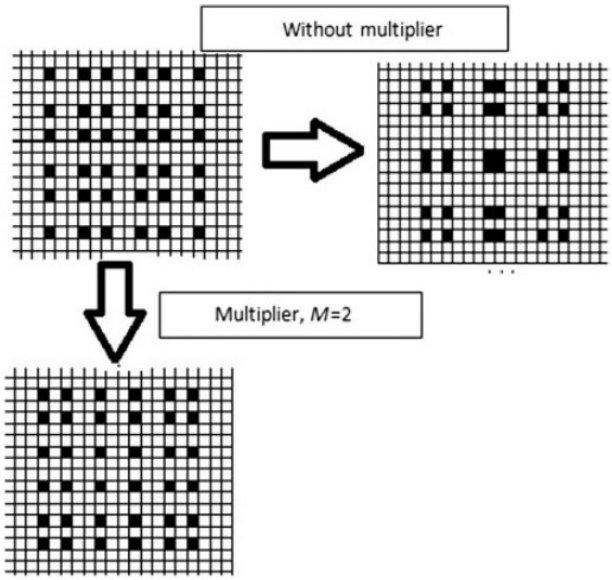

FIGURE 3.2: An example of a cycle and the usefulness of multiplication factor

We consider a network with communication radius 3 and the initial configuration consists of sensors in a 6 * 6 square. After some t time steps the configuration is depicted in Figure 3.2(a) (the cell marked as black contains a sensor) and in the next time step the configuration is as in Figure 3.2(b). It can be seen that the connectivity of the network is broken as shown in Figure 3.2(a). Hence the network applies the move back rules, and the resulting configuration is the same as Figure 3.2(a), which means that the network has entered into an infinite cycle. For this reason, the algorithm introduces a multiplier $M > 2$ that encourages a sensor to keep moving in its previous direction. The sensor (i.e. the state of the cell containing this sensor) remembers the previous movement direction and weights of neighbors in the opposite direction are multiplied with M. Going back to the example depicted in Figure 3.2, the computation step from configuration as shown in Figure 3.2(a) using a multiplier, $M = 2$, yields a configuration as in Figure 3.2(c).

We introduce rules that attempt to prevent the network from losing connectivity. Below we consider a movement step in the positive x-direction. The same rules apply to the three other directions. If a sensor s does not see any neighbors within distance $R_c - 1$ in the negative x-direction, a movement step in positive x-direction is blocked.

Consider a cell with m sensors $s1, s2, ..., sm$. Each sensor determines independently whether it should move or not. If all the sensors move to a positive x-direction, then sensor s_1 checks within distance $R_c - 1$ in the negative

x-direction for the connectivity and if there is no sensor within distance $R_c - 1$ then only sensor s_1 will not move to the positive x-direction. In case all sensors in the cell try to move in one of the other three directions, a similar control step is done in the opposite direction.

The above blocking conditions still do not guarantee the preservation of connectivity because the sensors that are within distance $R_c - 1$ of each other can move to the opposite directions as in Figure 3.3. For this reason, we introduce the following 'move back' rules. At any given time period t, before moving to the positive x-direction, a sensor s remembers whether or not it has a neighbor in two different quadrants (marked as circles Q_1 and Q_2 in Figure 3.4) in the negative x-direction. At the next time period $t+1$, if the sensor finds that there is no sensor in one of these quadrants but there was at least one sensor in the same quadrant at period t, then the sensor moves back in the negative x-direction. The same rule applies to three other directions.

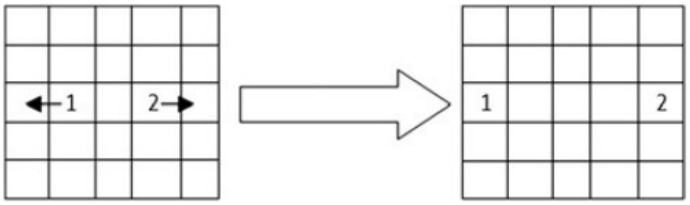

FIGURE 3.3: Neighbors break connectivity for Rc = 3.

If we are concerned only about the loss of connectivity, the move back rule can be simplified by remembering only whether or not there were neighbors in the direction opposite to the current movement, that is, in case of Figure 3.4, quadrants Q_1 and Q_2 can be combined together. We call this simplified rule the 1808-move back rule.

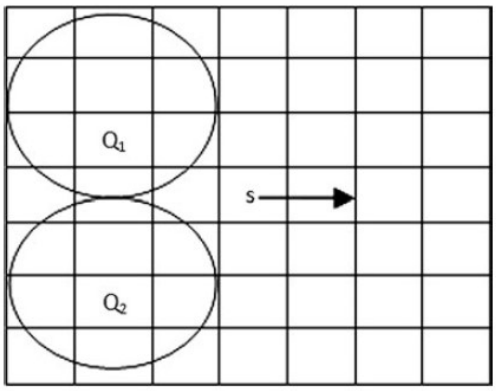

FIGURE 3.4: Quadrant Move Back Rule

If we have a cell with m sensors and x-move $(s_i) > 0$ for all $i = 1, ..., k$, where $k \leq m$, then in case of 180-move back rule, only sensor s_1 checks cells for a distance R_c in the negative direction and if it finds that there is no sensor in the negative direction then it moves back in the negative direction. All other sensors move according to the other rules. On the other hand, in the case of quadrant move back, before moving to a new direction each sensor remembers whether there is any sensor in the quadrants of the opposite direction.

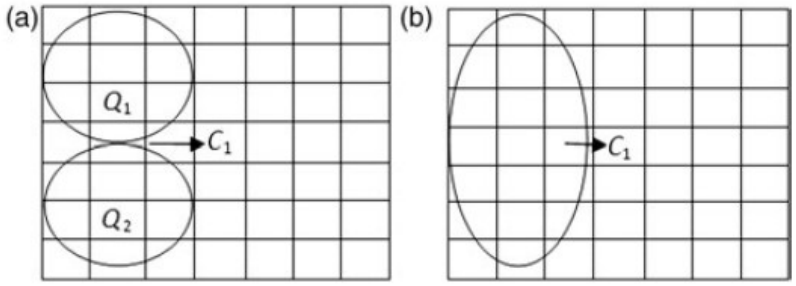

FIGURE 3.5: Move back rules for a cell with multiple sensors. (a) Quadrant move back and (b) 180-move back.

At any period t, s_1 checks the quadrants of the opposite direction and if it finds that there is no sensor in one of the quadrants at that time period but there was at least one in that quadrant at time $t - 1$, then only sensor s_1 moves back in negative x-direction. Other sensors continue with other rules. These rules are applied similarly in the other three directions.

For example, in Figure 3.5, a cell C_1 contains multiple sensors and all came from west then, in case of 180-move back rule sensor s_1 (one of the sensors in C_1) checks R_c distance in the west and in case of quadrant move back rule it checks Q_1 and Q_2 quadrants at the west.

Chapter 4

Proposed Method

In this chapter, we give a high-level description of the algorithm that disperses sensors from an initial configuration while trying to maintain connectivity in a hexagonal grid. In particular, for simplicity, below we talk about the movement of an individual sensor. However, in the general case, one cell may contain more than one sensor and the state of the cell needs to remember the information for each sensor it contains. We consider a scenario where $R_c > R_s$.

4.1 Mobile Dispersion Algorithm for Hexagonal Grid

Our initial algorithm is similar to the one introduced by Salim et. Al in [12]. The rules for moving each sensors of the network are discussed below. We also discuss rules to limit the movement of the nodes.

4.1.1 Local Rule for Defining the Movement of the Sensors

The basic rule is used to determine the direction a node will move. The movement each sensor is divided into two separate movements - movement along x-axis and movement along y-axis. To determine the direction of movement the weight of the neighbors must be calculated. The weights assigned to neighbors in case of $R_c = 3$ are as follows: the weights for neighbors at distances 1, 2 and 3 are 4, 2 and 1, respectively. In case of $R_c = 2$, the weights for neighbors at distances 1 and 2 are 2 and 1, respectively. The two axes for a sensor are shown in Figure 4.1

The movement along x-axis is determined by calculating the weight of the neighbors located to the left and right of the sensor. The sensor moves in the direction with lesser weight. If the weight on both sides are equal the sensor does not move along the x-axis.

The movement along the y-axis is calculated in a similar method to that of the x-axis. Here the weight of the neighbors located up and down of the sensor are calculated. The sensor moves in the direction with lesser weight. If the weight on both sides are equal the sensor does not move along the y-axis.

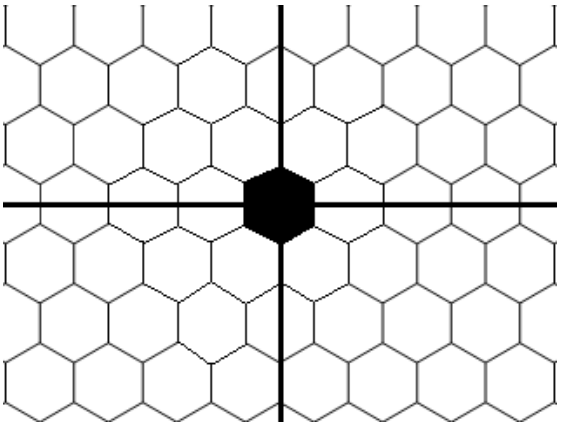

FIGURE 4.1: The x and y axes in the initial algorithm

4.1.2 Movement Blocking Rule

Movement blocking rule is applied to prevent disconnections in the network. If a sensor s does not see any neighbors within distance $R_c - 1$ in the negative x-direction, a movement step in positive x-direction is blocked. The same rules apply to the three other directions.

Consider a cell with m sensors $s1, s2, ..., sm$. Each sensor determines independently whether it should move or not. If all the sensors move to a positive x-direction, then sensor s_1 checks within distance $R_c - 1$ in the negative x-direction for the connectivity and if there is no sensor within distance $R_c - 1$ then only sensor s_1 will not move to the positive x-direction. In case all sensors in the cell try to move in one of the other three directions, a similar control step is done in the opposite direction.

4.1.3 Move Back Rule

The above blocking conditions still do not guarantee the preservation of connectivity because the sensors that are within distance $R_c - 1$ of each other can move to the opposite directions as in Figure 4.2.

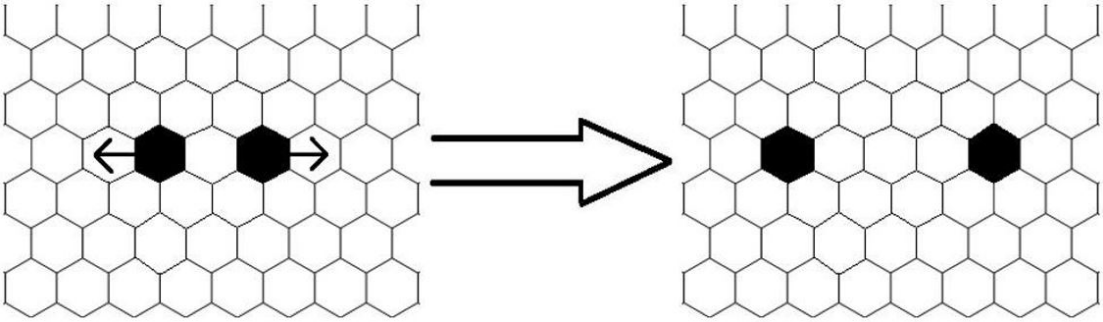

FIGURE 4.2: The x and y axes in the initial algorithm

The two black cells each contain one sensor each with $R_c = 2$. Calculating the weights of the left and right sensor determines that they must move left and right respectively. But such movement would mean that the sensors move out of each others communication range. As result a disconnection occurs in the network. This problem is eliminated by the application blocking rule.

For this reason, we introduce the following 'move back' rules. At any given time period t, before moving to the positive x-direction, a sensor s remembers whether or not it has a neighbor in two different quadrants (marked as circles Q_1 and Q_2 in Figure 3.4) in the negative x-direction. At the next time period $t + 1$, if the sensor finds that there is no sensor in one of these quadrants but there was at least one sensor in the same quadrant at period t, then the sensor moves back in the negative x-direction. The same rule applies to three other directions.

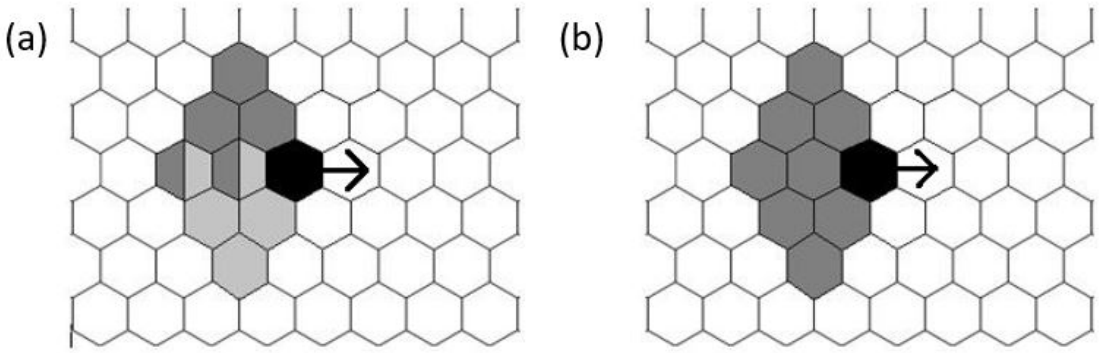

FIGURE 4.3: (a) Quadrant Move Back (b) 180 Move Back

If we have a cell with m sensors and x-move $(s_i) > 0$ for all $i = 1, ..., k$, where $k \leq m$, then in case of 180-move back rule, only sensor s_1 checks cells for a distance R_c in the negative direction and if it finds that there is no sensor in the negative direction then it moves back in the negative direction. All other sensors move according to the other rules. On the other hand, in the case of

quadrant move back, before moving to a new direction each sensor remembers whether there is any sensor in the quadrants of the opposite direction.

4.1.4 Movement Bound

Movement bound limits the total number of movement a sensor can make. A threshold value is selected. If a node tries to move more times than the threshold value its movement will be blocked. So the sensor will not move any further in the network.

Movement bound is used to conserve energy of the sensors. Movement requires a lot of energy for the sensors. So limiting the number of movement a node can make will save energy and increase the lifetime of the network.

4.2 Modified Algorithm for Hexagonal Grid

The algorithm that was initially proposed was a conversion of the method introduced in [12] with the addition of a simple movement bound rule. Although the performance of the algorithm was similar to that of [12] for some metrics, it faced some problems for some other ones. The most noticeable problem was the large number of disconnected sensor and the formation of loops. This problem arises because of the characteristics of the hexagonal grid. So some modification were required to be made to eradicate the problems. The modifications are discussed below.

4.2.1 Axes

The division of the axes needs to be changed for the hexagonal grid. The modified axes are shown in figure 4.4. The weight calculation of the up-down and left-right neighbors are done according to the new axes.

The change id made ensure that there is one cell at each radius distance along both of the axes. Also the number of cells on all four sides are equal in this orientation of the axes.

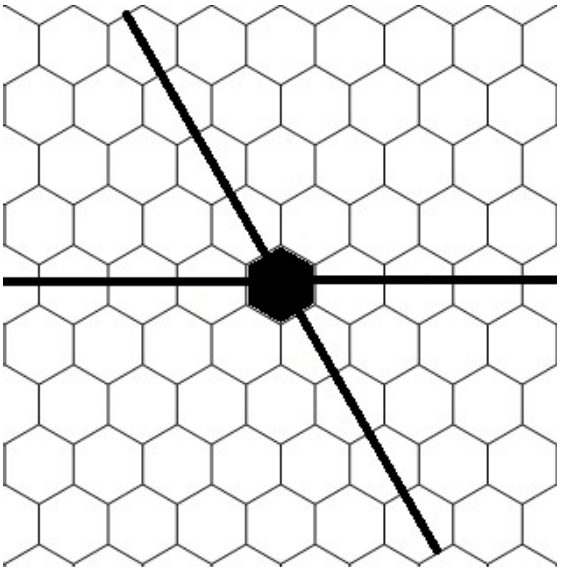

FIGURE 4.4: The x and y axes in a hexagonal grid

Also in case of hexagonal grid, for one of the axes the nodes will face some problem when it moves along that axis. In Figure 4.4 we can see that there is no position within radius 1 that is only up or down; they have component in either left or right also. When a node moves only up or down it must also move either left or right. In our algorithm we assume that when a node is required to move up or down, it choses the cell with the lower weight.

4.2.2 Move Back Rules

The 180 – move back rule faces some problems in case of hexagonal grid. This illustrated in Figure 4.5. The node 2 in Figure 4.5(a) calculated the weights and determines that it is required to move up and right. So it remembers that it had two neighbors to the left and two neighbors below. The position of the nodes in the next time step is shown in Figure 4.5(b). In this case if the node 2 apply 180 move back rule it determines that it does not require to move back and moves left. This creates a permanent disconnection in the network and the size of the largest connected component in the network is reduced from 4 to 2. So for hexagonal grid we only apply quadrant move back rule.

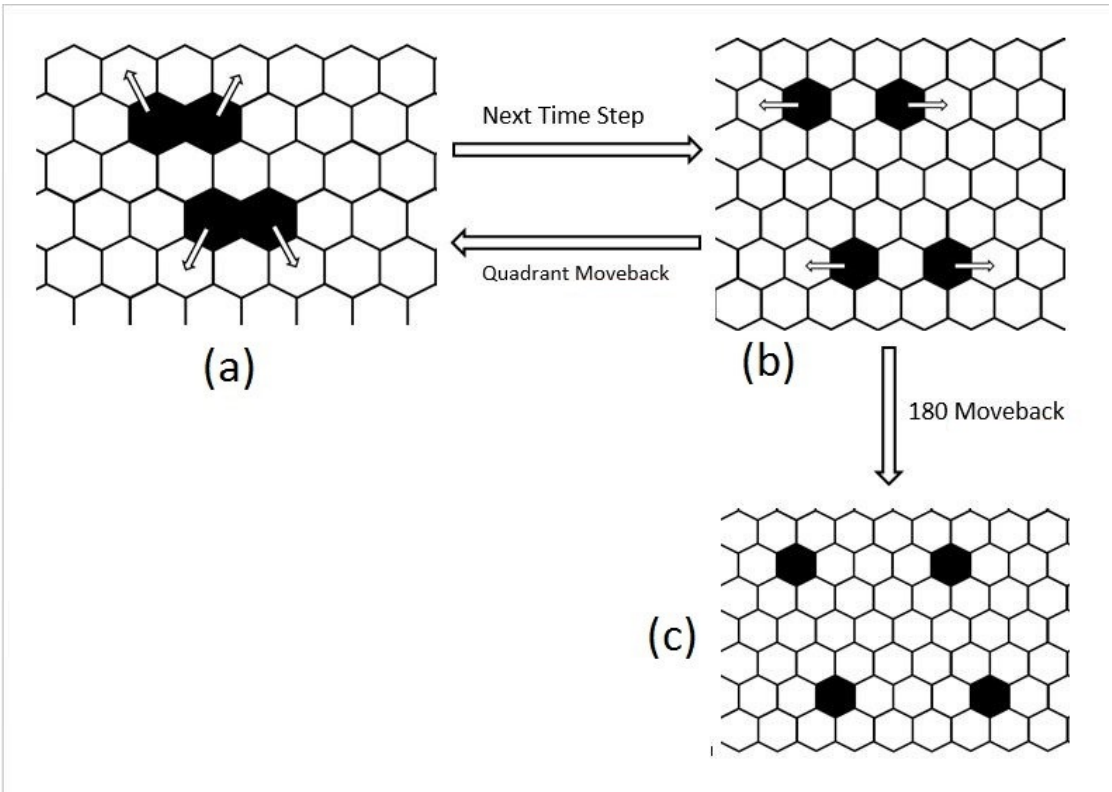

FIGURE 4.5: Limitations of 180 Moveback

To avoid this problem in hexagonal grid we will only use quadrant move back rule. The division of the two quadrants in case of hexagonal grid is shown in Figure 4.7. Such division is made to ensure equal number of neighbors in both quadrants. Here the dark grey color represents the first quadrant while the light grey color represents the second quadrant.

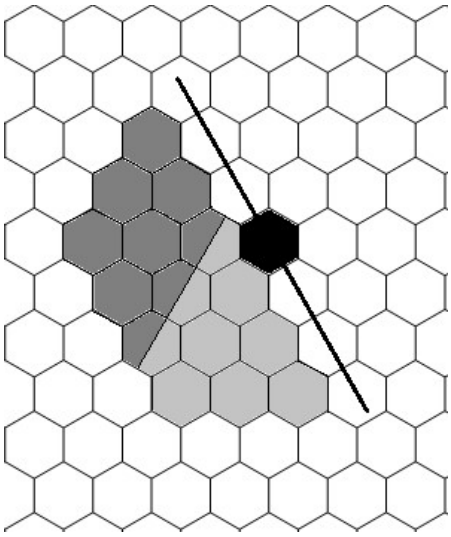

FIGURE 4.6: The two quadrants on the left side.

4.2.3 Application of a probabilistic model

Even after avoiding the 180 move back rule we face certain problems that occur irrespective of the choice of the move back rule applied. The number of nodes in the largest connected component remains too low for proper routing of data through the network.

The main reason for this problem is that the nodes simultaneously try to determine their next position in the grid. This problem can be solved if we move the sensors using a probabilistic model instead of a deterministic model. In the new method in each time step a node will try to move with probability p. If $p < 0.5$ then the node will move in that time step. Otherwise the node will do nothing. It will remain in its current position and wait for the next time step. The process will be repeated in all the subsequent time step.

The use probabilistic model greatly improves the disconnection problem. So this model is preferred in our algorithm over the deterministic model.

4.3 Movement bound

The energy of the nodes in the MWSN is very limited. Movement of the nodes require large amount of energy. To increase the lifetime of the sensors and consequently the network, the movement of the individual nodes should be restricted. Restricting the movement of the nodes helps improve the lifetime of the network at the cost of the coverage area. Balance should be ensured between the coverage and the lifetime of the MWSN.

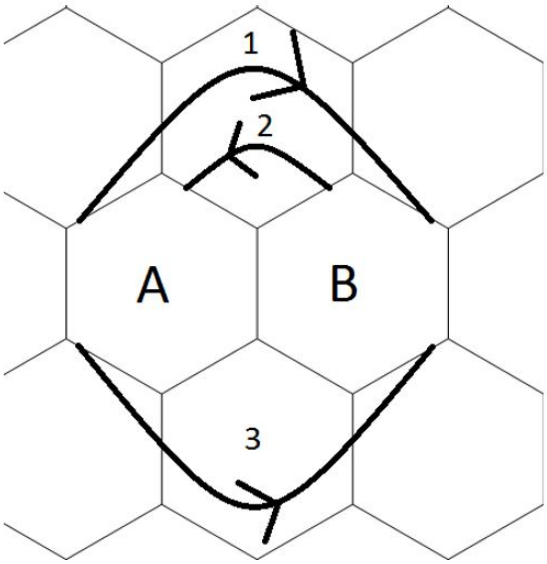

FIGURE 4.7: Movement Bound Rule

In our algorithm we have tried to stop the movement of a node if it fall in a loop. Suppose node n is at position A. During the next movement n moves to B. The node then returns backs to A. If for the next movement n wants to return back to B it is assumed that n has fallen in a loop and its movement is blocked. So if the four consecutive positions of a node is A-B-A-B then after returning back to A the node will not move any further.

4.4 Flowchart

The entire algorithm can be depictedd with the help of an algorithm as shown in Figure 4.8.

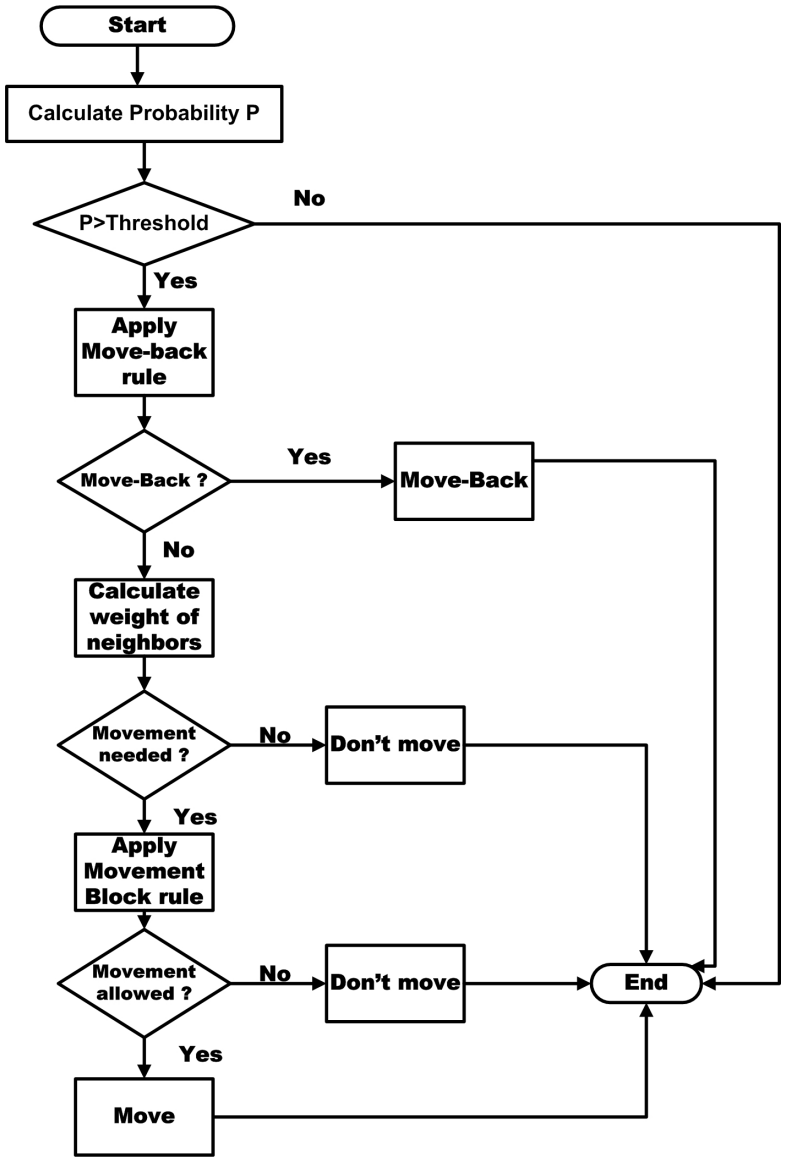

FIGURE 4.8: Flowchart of proposed mechanism.

Chapter 5

Simulation environment and results

5.1 Simulator

To test our CA-based algorithms we developed our own simulator. The codes of the simulator were written on Pyhton. The simulator determines the position of each of the deployed nodes and presents a graphical representation of the network for each time step. It also calculates the performance metrics which allows us to compare the performance of the different algorithms.

5.2 Simulation Environment

The results that are shown below are for networks containing 400 nodes. The nodes are placed in a hexagonal grid in two different distributions.The simulations were executed both with and without implementing the movement bound on the sensors. Since a probabilistic model was used there is a chance the simulation might continue up to infinity. To cope with such cases the maximum time step has been set to 1500 i.e. if the simulation continues for 1500 time steps it will terminate manually.

We considered two methods of deployment for the sensors. In the first method the 400 nodes were placed in a 20×20 grid where each cell contained exactly one sensor. In the second method the sensors were placed in a smaller 10×10 grid where each cell contain up to 5 nodes while certain cells may remain empty.

5.3 Performance Metrics

- Number of nodes in the largest connected component.
- Number of cells that are sensed.

- Strongly Connected Component (aSCC):

$$aSCC = \frac{averagecoverage(aCOV)}{averagehopdistance(aHD)}$$

- Average coverage

$$aCOV = \frac{totalcoverage}{numberofsensors}$$

  where total coverage means the number of cells that are sensed.

- Average hop distance(aHD)

$$aHD = \frac{\sum_{r1 \neq r2} hd(r_1, r_2)}{n(n-1)}$$

- hd(r_1,r_2) is the hop distance between r_1 and r_2

5.4 Performance Evaluation

The graphs in Figure 5.1 and 5.2 below showing the time step vs network coverage for three different value of probability threshold. If the threshold value is low the nodes will require more time step to reach its maximum coverage.

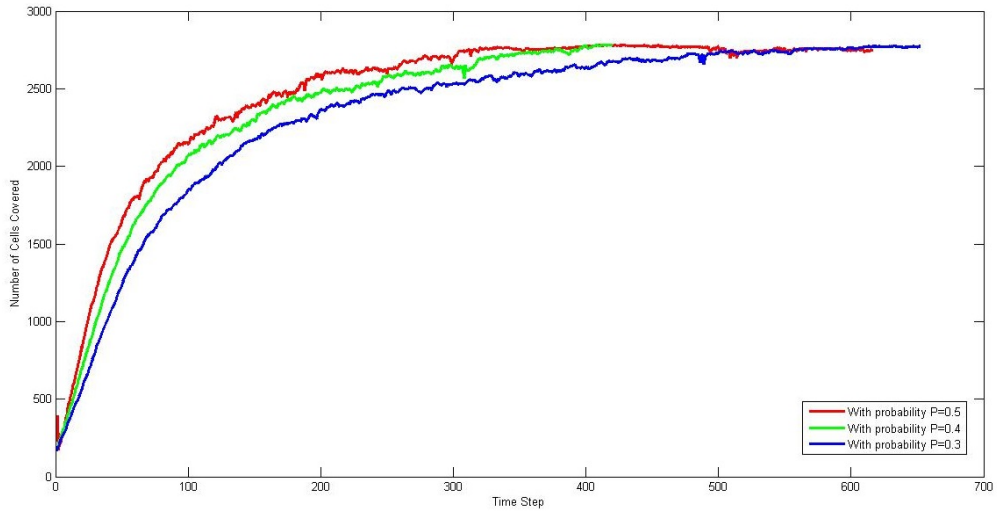

FIGURE 5.1: Performance Graph Without Movement bound

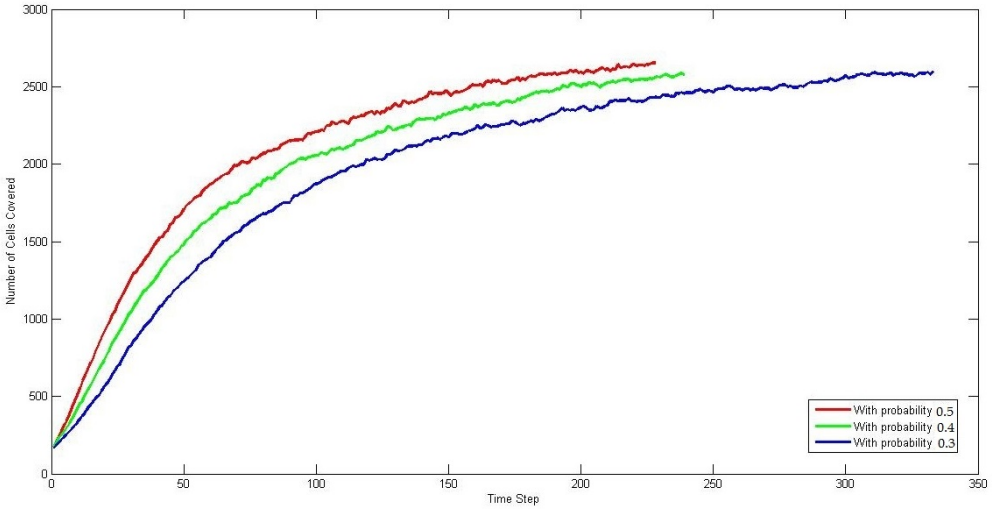

FIGURE 5.2: Performance Graph With Movement bound

Figure 5.1 and Figure 5.2 are showing difference between an algorithm using movement bound with an algorithm not using movement bound. Without movement bound the algorithm gives greater coverage, but the number of time steps is increased. That means it requires more movement which consumes a lot of energy.

On the other hand the algorithm using movement bound gives a little less coverage but it reduces movement dramatically. As movement consumes a lot of energy this movement bound algorithm offers a much more energy efficient solution at the cost of coverage. The loss of coverage is however insignificant compared to the reduction in movement of the sensors.

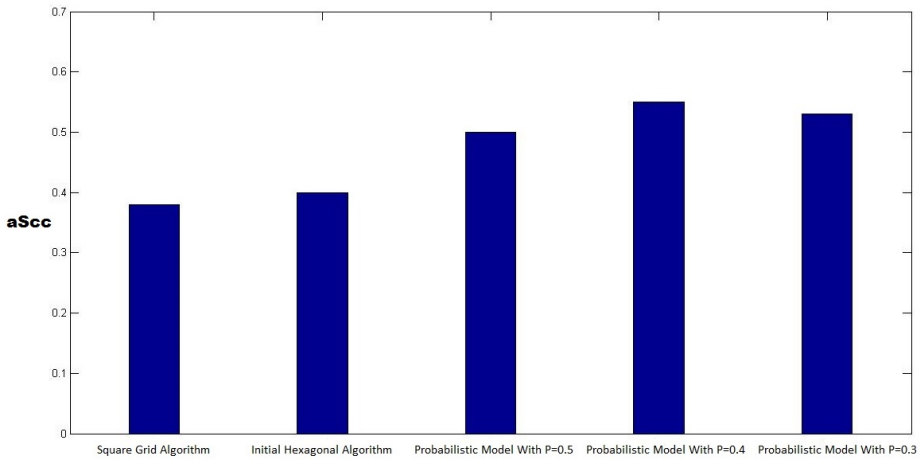

FIGURE 5.3: Different value of performance metric aSCC for different algorithm

Figure 5.3 shows the aSCC metric for the different algorithms. We see that the modified algorithm with a probabilistic model gives much better result compared to the other models. The data for the square grid was collected from [12]. Figure 5.4 shows the number of connected sensors in case of a deterministic model and a probabilistic model. The results shown here justifies the use of a probabilistic model over a deterministic model.

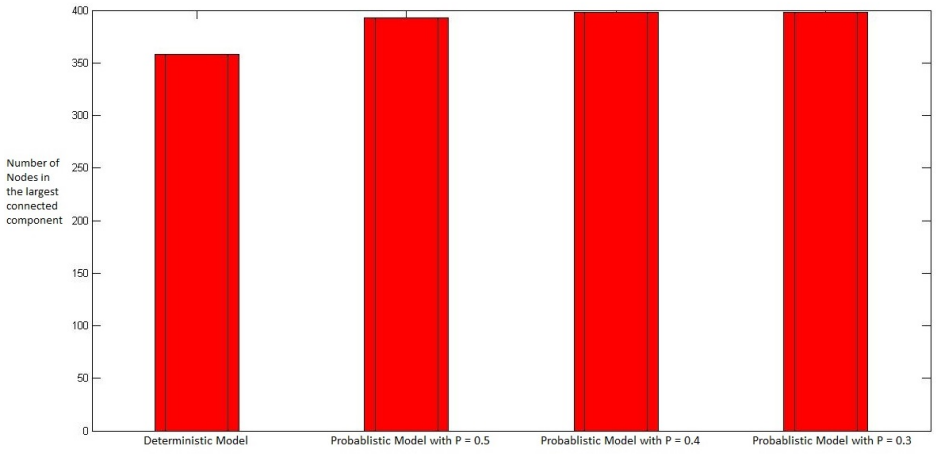

FIGURE 5.4: number of nodes connected for different algorithm

The tables shown below compare the minimum, maximum, average and median movements of the sensors with and without the application of movement bound. The results are shown for cases when the sensors are deployed deterministically also when they are deployed randomly.

TABLE 5.1: Results for 20*20 un-overlapped deployment without movement bound

time steps the simulation runs	probability	connected nodes	no of sensed cells	overlapped grid	max movement	min	avg	median
451	0.5	398	2786	0	159	32	94	101
409		398	2783	3	148	23	92	94
423		398	2779	7	147	20	92	95
434	0.4	399	2789	4	129	20	95	98
472		400	2799	1	152	28	92	97
550		398	2781	5	151	31	92	94
506	0.3	400	2797	3	157	29	95	95
397		394	2755	3	138	16	91	95
363		399	2786	7	157	20	92	94

TABLE 5.2: Results for 10*10 overlapped deployment without movement bound

time steps the simulation runs	probability	connected nodes	no of sensed cells	overlapped grid	max movement	min	avg	median
363	0.5	399	2786	6	157	20	91	94
528		394	2755	3	178	42	101	103
395		400	2793	7	158	44	94	96
454	0.4	394	2771	1	194	29	98	103
441		397	2773	6	156	26	96	95
559		400	2793	7	207	39	99	100
415	0.3	398	2782	4	152	30	94	95
353		400	2793	7	153	28	92	95
416		398	2781	5	142	30	93	95

TABLE 5.3: Results for 20*20 un-overlapped deployment with movement bound

time steps the simulation runs	probability	connected nodes	no of sensed cells	overlapped grid	max movement	min	avg	median
224	0.5	397	2624	155	11	0	7	7
236		397	2643	136	15	0	7	7
203		397	2583	196	14	0	7	6
207	0.4	400	2636	164	15	0	6	6
242		398	2646	140	9	0	7	7
245		398	2709	77	12	0	7	7
256	0.3	399	2705	88	11	0	7	7
222		400	2742	158	11	0	7	7
232		396	2646	126	9	0	7	7

TABLE 5.4: Results for 10*10 overlapped deployment with movement bound

time steps the simulation runs	probability	connected nodes	no of sensed cells	overlapped grid	max movement	min	avg	median
249	0.5	399	2646	147	18	0	8	8
225		398	2580	206	14	0	8	8
250		395	2629	139	19	0	8	8
248	0.4	398	2675	111	17	0	8	8
241		397	2638	141	17	0	8	8
250		398	2632	154	18	0	8	8
274	0.3	396	2699	73	16	0	8	8
212		398	2616	170	14	0	8	8
221		397	2627	152	18	0	8	8

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In our work we have developed an algorithm for CA-based MWSN for a hexagonal grid. The initial algorithm was converted from a similar algorithm applied in square grid. Along with the basic movement rules to increase the coverage there were some additional rules that improved the connectivity. A basic rule was also included to improve the lifetime of the network by limiting the number of move a sensor can make.

The initially proposed algorithm was based on an algorithm for square grid. So it did not address some of the uniqure characteristics of a hexagonal grid. Therefore some problems remained which were not addressed by this algorithm. So modifications were made to the existing algorithm to make it more suitable for the hexagonal grid.

One of the main changes that were made was the orientation of the axis. As the neighbors could be symmetrically divided following the square grid, the y -axis was tilted so that the division of the neighbors was accurate. The change in axes meant that the weight calculation had to be modified according to the new axes. Changes were also made to the quadrants in quadrant-move back rule to accommodate the new axes. The 180-move back rule had to dropped as it could not prevent disconnections in the network. The movement bound ruled was improved so that it could predict loops in the network and prevent them.

Despite the aforementioned changes in the network disconnections could not be prevented. To ensure no disconnections occurred a probabilistic model was introduced. Instead of trying to move in each time step the sensors now move according to a probability value.

In order to compare the performance the algorithms we developed a simulator using Pyhton. The Results from the simulation showed that our final

algorithms performed better that our initial algorithm also the algorithm developed for square grid.

6.2 Future Work

In the future we would like to further enhance our existing algorithm so that it can be used in a three dimensional space. Our current algorithm is limited a two dimensional plane and we would like to convert it to a algorithm for three dimensional space.

We want to develop sensor convergence algorithm where the distributed sensors will move back to a certain position in the environment. After the initial deployment the sensors disperse within the environment and collect environmental information. Usually the information collected by a single sensors is insignificant and the information of all sensors must be combined to make it meaningful. The sensors should therefore return to a specific position in the environment to collectively form meaningful information. We want to develop and algorithm to facilitate such movement.

Object tracking is another field that can be explored. After the sensors disperse they will track a certain object in the environment. Usually the object will be mobile and the sensors must communicate with other to accurately track the position of the object.

Bibliography

[1] G. Huang, "Casting the wireless sensor net", *Technol. Rev. (Manchester, NH) 106, pp. 50–57.*, 2003.

[2] B. Liu, P. Brass, O. Dousse, P. Nain, and D. Towsley, "Mobility improves coverage of sensor networks, in Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05", *University of Illinois at Urbana–Champaign, Urbana, IL, ACM, IL, USA, 2005, pp. 300–308.*

[3] S.A. Munir, B. Ren, W. Jiao, B. Wang, D. Xie, and J. Ma "Mobile wireless sensor network: Architecture and enabling technologies for ubiquitous computing", *in Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW '07), IEEE, Niagara Falls, Ontario, Canada, Vol. 2, 2007, pp. 113–120.*

[4] Banerjee, S. Das, H. Rahaman, and B. Sikdar, "CA based sensor node management scheme: An energy efficient approach, in International Conference on Wireless Communications", *Networking and Mobile Computing, 2007 (WiCom 2007), IEEE, hanghai, China, 2007, pp. 2795–2798.*

[5] Y. Baryshnikov, E. Coffman, and K. Kwak, "High performance sleep–wake sensor systems based on cyclic cellular automata", *in International Conference on Information Processing in Sensor Networks, 2008 (IPSN'08), IEEE, St. Louis, Missouri, USA, 2008, pp. 517–526.*

[6] S. Choudhury, K. Salomaa, and S.G. Akl, "A cellular automaton model for wireless sensor networks", *J. Cell. Autom 7 (2012), pp. 223–241*.

[7] R.O. Cunha, A.P. Silva, A.A.F. Loreiro, and L.B. Ruiz, "Simulating large wireless sensor networks using cellular automata", *in Proceedings of the 38th Annual Simulation Symposium, 2005, IEEE, Washington, DC, USA, 2005, pp. 323–330.*

[8] M. Esnaashari and M. Meybodi, "A cellular learning automata based clustering algorithm for wireless sensor networks", *Sens. Lett. 6 (2008), pp. 723–735.*

[9] M. Garzon, "Analysis of Cellular Automata and Neural Networks", *Texts in Theoretical Computer Science, Springer-Verlag, London, UK, 1995.*

[10] Sami Torbey, Selim G. Akl, "Reliable Node Placement in Wireless Sensor Networks Using Cellular Automata", *11th International Conference, WCNC 2012, Orléan, France, September 3-7, 2012.*

[11] Salimur Choudhury, Kai Salomaa, and Selim G. Akl, "Cellular Automaton Based Motion Planning Algorithms for Mobile Sensor Networks ", *In Proceedings of the First International Conference, TPNC 2012, Tarragona, Spain, October 2-4, 2012.*

[12] Salimur Choudhury, Kai Salomaa and Selim G. Akl, "Cellular Automaton Based Algorithms for the Dispersion of Mobile Wireless Sensor Networks ", *International Journal of Parallel, Emergent and Distributed Systems 2014, 29:2, 147-177*

[13] K.J.J. Kumar, K.C.K. Reddy, and S. Salivahanan, "Novel and efficient cellular automata based symmetric key encryption algorithm for wireless sensor networks ", *Int. J. Comput. Appl. 13(4) (2011), pp. 30–37.*

[14] K. Kwak, "Minimalist detection and counting protocols in wireless sensor networks ", *Columbia University, 2008, ISBN 9780549860716*

[15] W. Li, A. Zomaya, and A. Al-Jumaily, "Cellular automata based models of wireless sensor networks ", *in Proceedings of the 7th ACM International Symposium on Mobility Management and Wireless Access, ACM Digital Library, Tenerife, Canary Islands, Spain, 2009, pp. 1–6.*

[16] Sang-Ki Ko, Hwee Kim, and Yo-Sub Han "A CA Model for Target Tracking in Distributed Mobile Wireless Sensor Network ", *in Proceedings of the 13th International Conference on Control, Automation and Systems (ICCAS), Gwangju, Korea, 2013, pp. 1356 - 1361.*