MOBILE SENSOR MOVEMENT USING SIMULATED ANNEALING

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Sri Harsha Yamparala

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department
Computer Science

November 2009

Fargo, North Dakota

North Dakota State University
Graduate School

Title

MOBILE SENSOR MOVEMENT

USING SIMULATED ANNEALING

By

SRI HARSHA YAMPARALA

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

ABSTRACT

Yamparala, Sri Harsha, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, November 2009. Mobile Sensor movement using Simulated Annealing. Major Professor: Dr. Kendall Nygard.

Movement of mobile sensor nodes is an important aspect in wireless sensor networks, and there is considerable research underway into the design of mobile sensor movement algorithms for wireless sensor networks. Various mobile sensor movement algorithms have been proposed using different combinatorial optimization techniques such as Tabu Search, Ant Colony Optimization, and Genetic Algorithms. In this paper we present a mobile sensor movement Algorithm for Heterogenenous Sensor Networks using a Simulated annealing search technique. This paper also discusses the various factors that are taken into account while making a decision about the movement of a mobile sensor node in the event of a sensor node failure. With the help of Simulated annealing, a Movement Algorithm is proposed with an Annealing schedule that is suitable for Heterogeneous Sensor Networks, and its performance is evaluated for various sensor deployment scenarios in Heterogenenous Sensor Networks.

ACKNOWLEDGEMENTS

I would like to acknowledge the many people who made this paper possible: my adviser, Dr. Kendall Nygard, for his continued support, help, and direction; my family and friends for their editorial support and guidance; and everyone whose direction aided the completion of this project. My sincere thanks to Dr. Jung Kong, Dr. Gursimran Walia, Dr. Kendall Nygard, and Dr. Limin Zhang for serving on the committee.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

A wireless sensor network is made of sensor nodes distributed over a wide area and is used to co-operatively monitor various conditions. Some of these conditions could be physical or environmental, like temperature, sound, vibration, pressure, motion or pollutants. Wireless sensor networks are used in a wide variety of applications including battlefield surveillance, health care applications, environment and habitat monitoring, home automation and machine health monitoring. Movement options for the mobile wireless nodes in a sensor network are an important aspect to be considered in the event of failure of any sensor nodes in the wireless sensor network. This paper mainly presents a node movementalgorithm for heterogeneous sensor networks using the Simulated Annealing technique. We focus on the aspect as to which sensor is the most suitable one to replace another failed sensor. The algorithm that performs this Decision making works on the basis of the Simulated Annealing search technique [17].

Wireless sensor networks were initially made of homogeneous sensor type i.e. sensor nodes that have identical capabilities. But lately most of the Sensor networks are shifting from homogeneous to heterogeneous type. A heterogeneous sensor network consists of two or more types of sensor nodes (organized into hierarchical clusters). Some of the advantages that a heterogeneous sensor network possesses include increase in Network reliability and lifetime [19]. In a heterogeneous environment, the decision of movement a sensor node to the location of a failed sensor is not similar to homogeneous environment and involves consideration of various factors one of which is the tasks that were performed at that location by the failed sensor. Various other factors such as the range, power, and distance from the failed sensor should also be taken into account. Also it must be considered that these factors do not remain constant and vary throughout the network in a heterogeneous environment. Failure of nodes in such a distributed environment requires searching for a suitable replacement node on a global scale by taking the local conditions also into consideration. The necessity of considering many factors while moving mobile sensor nodes in the event of a sensor node failure in a heterogeneous sensor network makes the decision making process a complex task. Various algorithms were proposed such as [10][11] which laid more emphasis on providing coverage and broadcast capability in a heterogeneous environment. The algorithm proposed in [9] deals with waking up some sleeping nodes to perform some sensing functions when a particular event has happened in the sensor network. This algorithm might be considered in the event of node failures also but deploying a large number of nodes becomes expensive when the size of network increases as well as the type of nodes in a sensor network. The algorithm proposed in [8] lays more emphasis on maintaining the connectivity in the network and makes sure that power is utilized in an optimum way which in turn increases the reliability of the heterogeneous sensor network.

Given a heterogeneous sensor network which is made of both static and mobile nodes of different types and capabilities distributed over an area, the problem is to find a suitable mobile node in the event of a sensor node failure, which replaces the failed sensor node without much change in the overall sensor network parameters. The suitable mobile node has to take into account the various sensor network factors into consideration such as coverage and connectivity and it should not expend much of its own resources in replacing the failed node. In this paper, we present a mobile sensor movement algorithm that does the decision making process for moving the mobile nodes by taking all the required conditions such as coverage, connectivity, mobility cost and the tasks to be performed at the failed node location into consideration. The proposed algorithm is initiated whenever a sensor failure is detected in the network. The algorithm then starts the search for a suitable node for replacing the failed sensor and makes the computations required in the search process. The search process works on the principle of Simulated Annealing search technique. All the required factors are taken into consideration while computing a "solution value" for the sensor under consideration to replace the failed sensor node. The algorithm then executes the logic to find a suitable sensor for placing at the failed sensor location. If it does not find a suitable sensor in the first iteration of the algorithm, it then restarts the algorithm from the previous best solution. The important factors that are to be considered in simulated annealing are the annealing schedule (T, Tmax) and the good solution value (Smax).

1.1. Simulated Annealing

The basic approach taken for designing the algorithm lies with Simulated annealing [17]. Simulated annealing is a metaheuristic for global optimization problems that involves locating a good approximation to the global minimum of a given function in a large space. The name comes from a technique called "Annealing" used in metallurgy that involves heating and controlled cooling of materials to cause changes its physical properties. The following Pseudo code explains one of the basic forms of Simulated Annealing.

Pseudo Code for Simulated Annealing

```
        S ← S0; E←E(s)                    //Initial state, energy
        Sbest ← S; Ebest ← E                  // Initial "best" solution
         T ← 0; Tmax ← X       //Energy evaluation count and set Tmax value
while T < Tmax and E > Emax                   // While time left & not good
                                                 enough:
         Snew ← neighbour(S)                  // Pick some neighbour.
    Enew ← E (Snew)                         // Compute its energy.
if Enew < Ebest then                        // Is this a new best?
         Sbest ← Snew; Ebest ← Enew          // Save 'new neighbor' to 'best
                                                found'.
End if
 if P (E, Enew, temp (T/Tmax)) > random () then   // Should we move to it?
      S ← Snew; E ← Enew                    // Yes, change state.
      T ← T + 1                             // One more evaluation done
      goto while
End if
End while
return Sbest                            // Return the best solution found.
```

The process of Simulated Annealing starts with the initialization of the initial values of the algorithm namely initial state (S) and the initial energy (E). Then the best solution (Sbest) and the best energy state (Ebest) are given the initial values. Energy evaluation constant (T) is given the value zero and Tmax is assigned some value. The Annealing process then starts to evaluate each neighbor in the neighborhood set and compares it with the initial best solution (Ebest). If the newly found solution is the better than the existing solution then it updates the best solution (Ebest) with the existing solution. The procedure then proceeds to check if it has to change or move the initial state (S, E). This decision is based on the computation of the function "P(E, Enew, temp (T/Tmax))" and comparing it with a random number generator. If the function returns a value greater than the random number generator then the initial state is moved and the annealing process starts to evaluate the neighbors of that new state. When the annealing process comes to an end i.e. when its temperature schedule (Tmax) has been reached and when a solution has been found that is better than Emax has been found, the process then returns the best solution i.e. the Sbest it has found. Simulated Annealing is found to be applicable to a wide variety of optimization problems that include optimization of functions with both discreet variables as well as functions with continuous variables [16]. Also Simulated Annealing in found to be a promising algorithm in finding the global optimum of function and is found to be more robust than the conventional algorithms [6] and is less likely to fail for difficult functions.

1.2. Distribution of Sensors

The distribution of sensors in the field is assumed to be random. As mentioned in one the research papers [7], the heterogeneous sensor network under consideration here may not necessarily contain clustered areas, where sensors of one type are clustered together. When taking a large area into consideration, we assume that the sensor numbers of each type are increased uniformly.

1.3. Applications of Simulated Annealing

Simulated Annealing is one of the most popular optimization techniques and its applications extend to a wide range of areas including biology, geography, geology and various areas in the field of computer science; computational geometry in particular. One of the real time applications of Simulated Annealing include the design of IC Circuit boards [13][20], where the objective is to minimize the number of signals that must cross the chips while ensuring that there is will be room left on the chips by attempting to evenly distribute the number of circuits. In the field of Image correction, Simulated Annealing is applied to image generation for high resolution computer graphics using Positron Emission Tomography (PET) [20]. Also Simulated Annealing has found various applications in the form of hide and seek algorithm which is nothing but a continuous simulated annealing method. This method is used for various optimization problems such as Yield Optimization [18]. A quick search through the Engineering and Computer Science literature will return almost 300 papers with "simulated annealing" in the title.

1.4. Annealing Schedule

An essential feature of Simulated Annealing is its annealing schedule. As the simulation proceeds the temperature is gradually reduced. The temperature is initially set to a high value at the starting stage of the simulation and then is gradually reduced at each step according to some annealing schedule- which may be specified by the user but at the end of the simulation the temperature must be zero. In this way we make sure that the system has wandered around a broad region of search space containing good solutions before reaching the best solution at the end of the annealing schedule. The Annealing schedule for the Simulated Annealing is decided by the parameters T and Tmax. While at the start of the simulation, the temperature is set to Tmax and for each step in simulation it is decreased according to the annealing schedule. At the end of the simulation, the value of T must be zero. It has also been shown that the probability the Simulated Annealing algorithm terminates with a global optimum solution approaches 1 as the annealing schedule is extended [21]. This may not be acceptable in many cases as the objective of any search procedure is to find the best solution in a certain amount of time. Various annealing schedules have been proposed like exponential temperature schedule and non-linear temperature schedules i.e. temperature schedules with different temperature decrements.

1.5. Parameters in Simulated Annealing

In addition to the Temperature parameters (T and Tmax), Good enough solution (Emax) and Probability of moving to a new solution (Pmov) are the other parameters that influence the way Simulated Annealing works. The parameter "Emax" decides if the newly found solution is a good enough solution for the simulation or not. This parameter is used to decide if at the end of the annealing schedule, the search process has found a solution that is good enough for terminating the search process. If a best solution has not been found that is equal to or better than the "Emax" value then the search process continues to evaluate the neighborhood candidates. The parameter "Pmov" decides if the newly found solution is a good enough solution to be considered the initial solution of the simulation. This is decision is made based on the values of other parameters like T, Tmax, E and Enew. So the probability that the recently computed solution (Enew) will be the initial solution (E) of the simulation will also be based on the temperature parameters (T and Tmax). So if a solution has been made the new initial solution, then that does not necessarily mean that it is the best solution found so far in the simulation. As mentioned before, the decision made also depends on the temperature T at that time of Simulation. The probability of moving to the new initial solution is then compared to a random number generator to make a decision as to assign the newly found solution as the initial solution of the simulation. The simulation then starts to evaluate the neighbors of the new initial solution (E).

1.6. <u>Research Goal</u>

The main focus of this paper is to design a mobile sensor movement algorithm using Simulated Annealing technique for the heterogeneous sensor network that does the decision making required for moving the sensors in the event of a node failure. In our sensor network model, the sensor nodes i.e. both static and mobile are distributed at random. In a heterogeneous environment, where static nodes of different types are distributed at random we would like to test for various failure patterns and observe the changes in the sensor node placement, coverage and connectivity of network. The overall objectives of this paper can be stated as follows:

1.) To design a mobile sensor movement algorithm for a heterogeneous sensor network using Simulated Annealing technique that takes various factors such as network connectivity, coverage, mobility cost and local conditions such as the failed node tasks into consideration.

2.) Validate the various aspects of the algorithm like Temperature schedule, neighborhood search and best solution found for different heterogeneous environments.

3.) Test the scalability factor for the algorithm and determine the criteria for the designing the algorithm for large sensor networks.

4.) Test the behavior of the algorithm for complex environments and determine the limitations of the algorithm for complex environments.

1.7. Preliminaries

This paper considers that all sensors know about their locations and all the mobile nodes have the same range. The algorithm is then initiated whenever a filed sensor node is detected in the sensor network. Communication between the sensor nodes is not considered in this paper and it is assumed that sensors of different types in the network have some communication interface through which they can relay or transmit the network information.

1.8. Paper Organization

The remainder of the paper is organized as follows: Chapter 2 discusses the various on going research regarding mobile sensor movement algorithms. In Chapter 3, the Simulated Annealing model is presented which was used for designing the mobile sensor movement algorithm for the sensor nodes in the heterogeneous sensor networks. In chapter 4, the experimental work is presented and the results obtained from the various types of experiments that were conducted. In Chapter 5, we provide our conclusions and the future work.

CHAPTER 2

DISCUSSION OF PREVIOUS ALGORITHMS

There is much ongoing research in the area wireless sensor networks; regarding placement of sensors, deployment, tracking, cost calculation, etc. In this paper we concentrate mostly on the research that is going on in the placement of sensor nodes i.e. movement of mobile sensor nodes in a wireless sensor network and provide an overview of traditional approaches. We also discuss about some of the advantages of our approach compared with the traditional approaches.

2.1. Neural Networks

Neural networks are used for a wide range of applications in wireless sensor networks including localization techniques, data classification, data integration, movement of sensor nodes, broadcast scheduling and so on. An example of a Neural Network that performs decision making is the perceptron. Perceptron consists of artificial neurons arranged in layers. A simple perceptron consists of 3 layers – an input layer, a hidden layer and an output layer. The signal is fed form the input layer to the hidden layer which in turn feeds the signal to the output layer. Perceptron is considered to perform decision making by recognizing and classifying information. By having a complex wiring of neurons in between layers and by having additional layers in between the input layer and the output layer, it is possible to perform some classification and recognition of the input signals and perform some decision making. These decision making principles have found wide range of applications in wireless sensor networks which in turn led to the design of various mobile sensor movement algorithms. The main difference between our approach i.e.

Simulated Annealing and Neural Networks is that the latter learns how to approximate a function while Simulated Annealing searches for global optimum. This gives neural networks a huge advantage in modeling changing environments [17].

2.2. <u>Genetic Algorithms</u>

Genetic algorithm is a search technique used to find exact or approximated solutions to optimization or search problems. A Genetic algorithm usually starts by working on an initial population which is randomly generated. The initial population could be a large varying from hundreds or thousands of solutions covering the entire search space. The algorithm starts to work on the initial population by applying the fitness function and selecting only the best-fit solutions for crossover or mutation. This process generates a new set of solutions which replace the old solutions. This process of mutation is done until a minimum criterion is met or a fixed number of generations have reached or the time has expired. Genetic algorithms have been compared with an enhanced version of Simulated Annealing called Adaptive Simulated Annealing (ASA) and is shown that ASA outperforms Genetic algorithms in every case [17]. Also there seems to be that Genetic algorithms offer no statistical guarantee that the global minimum will converge to an optimum point [17]. In some particular applications such as in biology [4], genetic algorithms are found to provide better solutions than Simulated Annealing.

2.3. <u>Gradient Methods</u>

Gradient methods usually work on optimizing well behaved continuous functions, by relying on the information about the gradient of function to guide the search. There are two types of Gradient search algorithms namely Gradient descent and Gradient ascent algorithms. Gradient descent algorithm is used to find the local minimum of function by taking steps proportional to the negative of the gradient of the function at the current point. The step size 'dx' plays an important role in deciding the performance of the gradient descent algorithm. Gradient descent algorithm suffers from the drawbacks that it can take many iterations for the algorithm to converge to a local minimum of the curvature is different in different sections. Also, finding the optimal step size can be time consuming and a fixed step size can yield poor results. On the other hand, the gradient ascent algorithm suffers from the drawback that it can perform well only on unimodal functions i.e. functions that have only one peak [17]. For multimodal functions, the gradient ascent algorithm reaches the first peak and this may not be the highest peak. After making to the top of the first peak, no further progress is made.

2.4. Iterated Search

Random search and gradient search can be combined to give an iterated hill climbing search. Iterated search initially proceeds in a pure hill climbing way and after it has reached a peak the process is started all over again from another randomly chosen start point. This technique is known for its simplicity and is found to perform well if the function does not have too many local maxima [17]. The disadvantage of this technique is that each random trial is carried out in isolation and no overall picture or shape of the domain is obtained. Iterated search technique is found to outperform both Genetic algorithms and Simulated Annealing in scenarios where the maximum is located in a small region surrounded on all sides by regions of low fitness.

2.5. Art Gallery Problem

There are many issues that have to be taken care of when considering movement of mobile sensor nodes and the art gallery problem mentions a few of them. Art gallery problem deals with the issue of how many sensor nodes are required so that each point in the sensor network is seen by at least one sensor node. While at the first instance this might seems to be coverage or connectivity related issue, but when dealing with a sensor network which may be homogeneous or heterogeneous, the coverage and connectivity are important factors to be considered while taking a decision regarding movement of mobile sensor nodes where uniform coverage might be an important factor . The art gallery problem begins by considering the total area as a polygon and by decomposing the polygon into smaller polygon and by using one of the vertices to cover the wall or the edge of the polygon and finding the dual of the current polygon. One of the applications of the art gallery problem is in the placement of the cameras for sports event which require monitoring of certain events.

CHAPTER 3

SIMULATED ANNEALING MODEL

In this chapter, we present the Simulated Annealing model that was used for designing the mobile sensor movement algorithm for the heterogeneous sensor network. We also discuss about the various measures that were taken to prevent the algorithm from being trapped in the local minima and also discuss early termination of the algorithm once a good solution has been found at the initial stages of the annealing schedule.

3.1. Annealing Schedule

As mentioned before, the annealing schedule plays an important role in finding a global optimum solution in the search space of the Simulated Annealing algorithm. The parameters “T” and “Tmax” determine the number of steps that the Simulated Annealing algorithm will take in the search space in finding the global optimum solution. The temperature decrement “dT” also plays a major role in deciding how much the temperature will be decremented after the evaluation of each candidate in the search space. Parameter “dT” has to chosen in a way such that at any point in the search process when a global optimum solution has been found the search process should not waste time evaluating the rest of the candidates in the search space and should terminate the search quickly and return the global optimum solution. A point to note is that when a global optimum solution has been found, the search technique must still evaluate a few candidates in the remaining candidate set before the search is terminated. For this reason, we take two temperature decrements “dT1” and “dT2” in our search technique. dT1 is the optimum temperature decrement where the Simulated Annealing algorithm evaluates every candidate in the candidate set to find the global optimum solution. "dT2" is referred to as the accelerated temperature decrement where the algorithm terminates after evaluating at least half of the remaining candidates in the candidate set, when a global optimum solution is found. At the initial stages of the search technique, the parameters "dT1" and "Tmax" are assigned values making sure that the algorithm evaluates every candidate in the search space. During the starting point of the algorithm, the temperature decrement "dT" is set to "dT1". Whenever a global optimum solution has been found during the search process, the algorithms temperature decrement "dT" is set to accelerated temperature decrement "dT2" and the algorithm terminates after evaluating at least half of the remaining candidates in the search space. At the end of the annealing schedule, if the best found solution is less than the good enough solution "Emax", then the algorithm is restarted from the previous best known state with an annealing schedule slightly less than the previous iteration of the search process. This is also known as restarting of the algorithm and is useful in finding the global optimum value of a function that is discontinuous or undergoes rapid changes in its values.

3.2. Parameters of Simulated Annealing

The other parameters that influence the way the search technique proceeds are the good enough solution "Emax" and the probability of moving to a new solution "Pmov". In this algorithm, the value of "Emax" is set slightly higher than the average of the set of solution values encountered in the search process. The value of the "Emax" decides when the search should end. If at the end of the temperature schedule, the best solution value is less than the value of Emax then the search process continues to evaluate candidates with a new annealing schedule. In any search technique, proper care has to be taken to make sure that the search process does not get stuck in the local minima. In simulated annealing technique, to avoid the local minima problem, a new initial solution has to be taken every time after the algorithm has evaluated a certain number of candidates. Thus the value of "Pmov" has to be computed in a way that even if the current solution "Enew" is a bad solution when compared to the initial solution "E" for some temperature value "T" , the current solution must be set to the new solution "Enew". Therefore in order to avoid the local minima, the Simulated Annealing algorithm must take a couple of bad moves in the search process.

Thus the movement of the initial solution E necessarily does not have to follow the best solution path and also include a couple of bad moves, in order to avoid being stuck in the local minima. In this Simulated Annealing model, the value of probability of moving to the next solution "P(E, Enew, temp (T/Tmax))" is computed in a way that the initial solution path proceeds in the best solution path for some extent in the annealing schedule and as the annealing schedule is mid way or is coming to an end, the initial solution path takes bad moves so as to maximize the chances of finding a global optimum solution.

3.3. <u>Pseudo Code for Simulated Annealing Model</u>

In this section, we present the pseudo code for the Simulated Annealing model that we used for designing the mobile sensor movement algorithm for heterogeneous sensor networks. Various aspects such as restarting of the algorithm, avoiding the local minima and accelerating the annealing schedule when a good enough solution has been found in the search process have implemented in our Simulated Annealing model. Since this model is being used for evaluation of the sensor parameters and computing its solution by taking its parameters, there is a slight change in terminology in the new model. In the basic Simulated Annealing model, we pick a neighbor from a neighborhood set. In the new model, this is usually referred to as picking a set of sensor co-ordinates, from the neighborhood set. Sensor co-ordinates usually refer to the x-coordinate, y co-ordinate and the range 'r' of the sensor. The term "computing the energy" in the basic Simulated Annealing procedure is analogous to the term "computing the objective function" of the sensor node. The objective function of a sensor node is basically a value computed by taking various factors into consideration such as the Euclidean distance of the sensor node from the failed sensor, the mobility cost, the functional specification of the sensor under consideration and the range of the sensor. The good enough solution "Emax" in the basic model is referred to as "Smax" in the new model.

Pseudo Code of the Simulated Annealing Model

```
Initialize the parameters T, Tmax, dT1, dT2, xprev, yprev, rprev, Sbest, Smax, Rcount.
    Fetch the failed sensor co-ordinates (xf, yf, rf)
  If xprev, yprev, rprev is nothing then
        Get the neighborhood set of the failed sensor co-ordinates (xf, yf, rf)
  Else
        Get the neighborhood set of the (xprev, yprev)
  End if
  If Rcount equals zero then
        Set Sbest, Tmax and dT values to the normal annealing schedule parameters
    Else
        Set Smax to a value less than previous value and Tmax to half the maximum
        temperature annealing value.
    End if
For each (x, y, r) in neighborhood set
        Calculate the objective function or the solution value S(x, y, r)
If S >Sbest then
        Update the new best solution value Sbest and the new best state (xbest, ybest, rbest)
End if
If Sbest >= (Smax + (0.1) * (Smax)) Then
            dT = dT2    //Set the Temperature decrement to accelerated decrement value
End If
If Pmov(S, Snew, T, Tmax)> random number
              Set the initial state and the initial solution to the new solution and new state
              values.
End if
    T=T-dT                          //decrement the temperature

End for
If Sbest <Smax then
            If Rcount not equals 1 then
                    Rcount=Rcount +1
            End if
            If xp, yp, rp= nothing then
                  Restart the process with previous sensor co-ordinates set to nothing
              Else
                  Restart the process with the previous co-ordinates
          End if
End if ;
 Return the best state values xbest, ybest, rbest
```

The output values for the new Simulated Annealing model would be the global optimum for the parameters x, y, r. At the starting point of the algorithm, the initial algorithm values such as xi, yi, ri, Sbest, Smax, Rcount, T, Tmax, dT1, dT2, xprev, yprev, and rprev are set to their initial values. The search for the global optimum value of the sensors starts with fetching the neighborhood set of the failed sensor node. The algorithm then proceeds to compute the solution value for each sensor in the neighborhood set. The solution is then compared with the best solution value "Sbest" and if the existing solution is better than the existing best solution value , then the best state values xbest, ybest, rbest and the best state value and the best solution values are overwritten by the existing state xu, yu, ru values and the existing solution value Snew. Thus in this way the algorithm follows the best state in evaluating the neighborhood of the initial solution. Therefore if the initial solution is not moved from the point where it was at the starting of the algorithm, then the algorithm returns the best state and solution for that neighborhood. In the case of complex environments where the solution value might undergo rapid changes or is discontinuous then the algorithm might be stuck in local minima.

To avoid being stuck in the local minima, the function "Pmov(S, Snew, T, Tmax)" is computed in a way such that the initial solution is path might make some bad moves. A bad move is considered as moving an initial solution to a new solution which may not be a better one for some T and Tmax values. The function Pmov(S, Snew, T, Tmax) is computed in a way that the initial solution path follows the hill climbing strategy at the starting point of the algorithm and as the annealing schedule is mid way and is coming to an end, the function that computed the parameter "Pmov" returns values that forces the initial solution path to make bad moves so that the algorithm can span the search space to find the global optimum solution. This helps in avoiding the local minima problem in the algorithm and ensuring that the search technique covers the entire search space. Also a multi-valued temperature scheduling makes sure that the search process does not wander through the search space too much even after a fair enough solution has been found. To make sure of this, a two level temperature scheduling is implemented. The value of the accelerated temperature decrement "dT2" is usually set to 50 percent more than the optimum temperature schedule "dT1". The value of dT1 and Tmax are chosen in a way so that the entire search space is covered under normal conditions and the initial temperature decrement "dT" is set to "dT1". Also whenever a solution is encountered that is at least 10% better than the good enough solution (Smax), the annealing schedule is decremented in an accelerated way. At the end of the search process if we do not find a solution that is at least as good as Smax then we restart the algorithm with the previous best state set as the initial state for the algorithm and the best solution value as the current best solution. The algorithm then proceeds to evaluate the neighborhood of the current solution in search for the global optimum value. Also the algorithm is designed to restart only once and if no solution even after restarting the algorithm, then the algorithm returns the most previous best solution values.

CHAPTER 4

EXPERIMENTAL WORK

This chapter explains about the experimental work that was carried out in heterogeneous sensor networks. At first, an introduction to the Simulation interface is given and screenshots are provided to give an idea about initial and final stages of the algorithm. Later, the experimental work that performed on the two sensor network models is presented and the results are discussed.

4.1. Simulation Interface

The Simulation interface is designed as a Windows application and is written in .NET programming language. The graphical content of this interface is designed using Graphics Device Interface (GDI+) and the simulation proceeds one at time i.e. the complete simulation is carried out in one iteration. The data about the sensors is stored in a database and information about the sensors is retrieved when the simulation starts and is updated periodically during the simulation process. Different types of sensors are drawn in different colors. In this Simulation, static sensors are drawn in blue on the canvas and mobile sensors are drawn in black. The co-ordinates of the Simulation area start at (0, 0) and is located at the top left hand corner of the screen. The co-ordinates of the sensors are taken with respect to the top left hand corner of the screen. Information about the sensors is retrieved using the data access layer of the .NET programming language. This data access layer is also referred to as the Active Data Objects (ADO) in .NET programming. Figure 4.1 gives a representation of the initial distribution of both the static sensors and mobile sensors before the failure of sensor nodes and the algorithm is initiated.

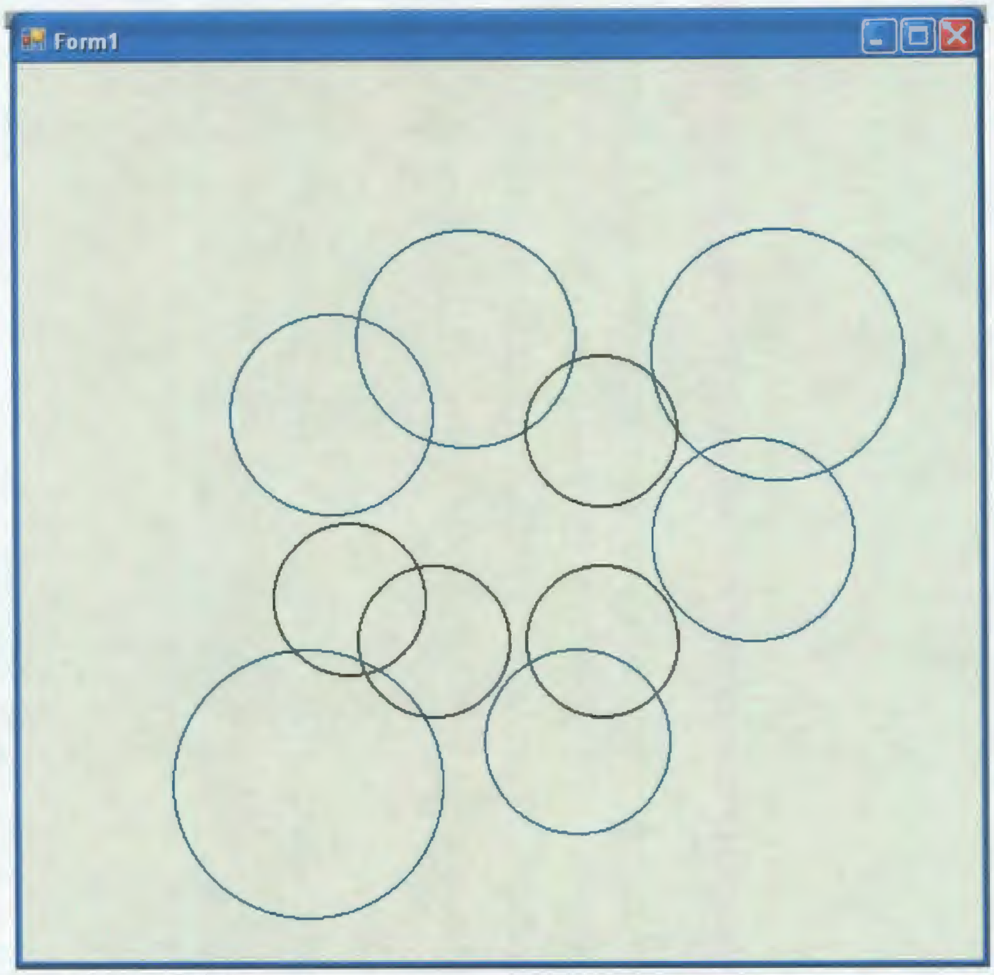


Figure 4.1. Initial Setup of Sensors.

The co-ordinates of the static sensors starting from the top left hand corner of the screen and going clockwise are (125,150), (200,100), (375,100), (375,225), (275,350) and (90,350). The co-ordinates of the mobile sensors starting from the top of the screen and going clockwise are (300,175), (300,300), (200,300) and (150,275). In all our experiments all failures of sensors are considered to be static and mobile sensors are considered as candidates for the replacement of the failed sensors. For showing the failure of sensors and their replacements, different colors are used for each of the sensor node failure and its corresponding replacement. A static node failure is shown in a black color and the initial and final positions of the replacement mobile node are shown in the same color.

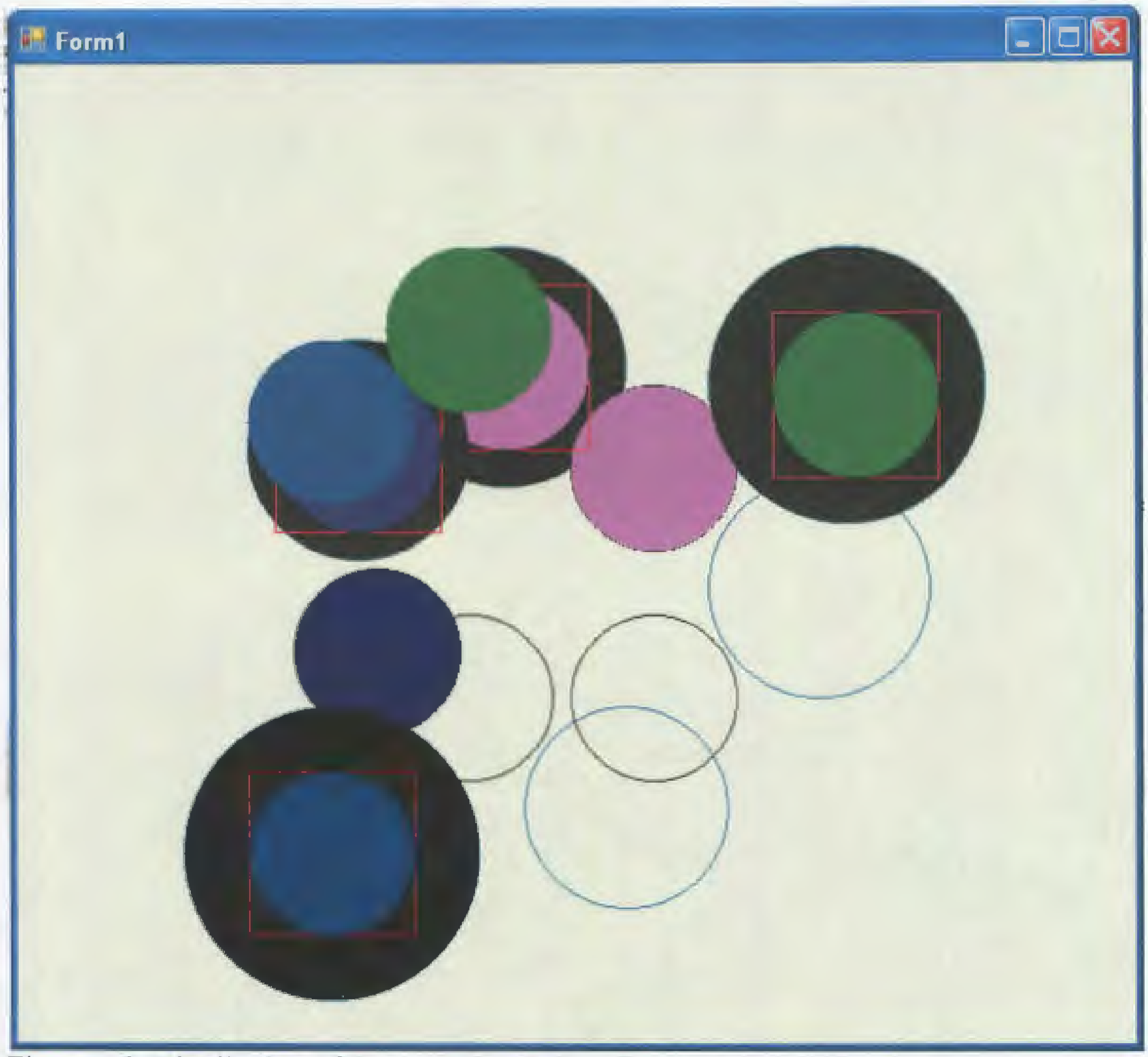


Figure 4.2. Distribution of Sensors after Failure Pattern.

The above figure shows the distribution of the sensor nodes after a series of static sensor node failures. As mentioned before the failure of a static sensor is shown in black color. The final position of a mobile sensor is shown as a red box and the initial position is shown in the same color as it was in the final position. In the simulation above, the sensor network is of heterogeneous type and has 4 types of sensor nodes. The tasks that each sensor node performs in the sensor network are stored in the database. For every failure of static sensor node, these tasks list are taken for finding a suitable replacement among the mobile nodes. The decision as to move which of the sensor nodes is a complex process and involves computing various parameters. Because of these reasons, in the simulation above even though 4 static sensors failed it resulted in movement of only 2 mobile nodes. Also the order in which sensor node replacements took place are denoted in the colors as Violet, Indigo, blue and green. Therefore a sensor node replacement which is of violet color is the first to happen in the series of failures and the sensor node replacement denoted by green color is the last to happen.

4.2. Working of the Simulation

In each of our simulations, a series of sensor node failures are considered and for each of the sensor node failure a call to the decision making algorithm or the mobile sensor movement algorithm is made. The mobile sensor movement algorithm then takes the failed sensor nodes parameters such as its (x, y, r) co-ordinates and its functional set. The functional set comprises of the various tasks that the failed sensor node used to perform at its location and is used in evaluating various other mobile sensor nodes that might be used as a replacement for the failed sensor node. The pseudo code for each of our simulations is presented below.

Pseudo Code for a Simulation

```
For each sensor node failure in the failure pattern
        Execute Decision Making Algorithm with failed sensor parameters
        Move the best suitable sensor to the failed sensor location
        Update the moved mobile sensor co-ordinates in the database
    Next;
```

```
        End For
```

The above pseudo code is for a single simulation and many simulations were carried out on sensor network models.

4.2.1. Computation of Solution Value from Sensor Parameters

Solution values are computed for each of the sensor node in the process of the decision making algorithm. Various metrics exist for the sensor network platforms depending on the application and architecture of the sensor network and the resources available. A generalized set of metrics were proposed in [12], of which power consumption and transmission capacity are found to be problematic. For this reason, we do not take these factors into consideration while computing the solution values. The solution values are computed based on the sensor nodes location co-ordinates (x, y), its radius 'r' and its functional set which comprises of the sensor nodes functional tasks. Coverage and Connectivity factors are determined from these sensor parameters. The location co-ordinates of the static sensors are retrieved from the database during the initial phase of the simulation and stored in the algorithm variables. For retrieving other sensor node parameters such as the mobile sensor location co-ordinates and its functional set, the algorithm has to make a call to the database. During the process of computing the solution values, the algorithm computes the Euclidean distance between the failed sensor node and the sensor under consideration, the total mobility cost for moving the mobile node to the failed sensor location, functional convergence or correlation, coverage gain with respect to the failed static sensor and connectivity of the mobile node at the failed sensor location. Different weights are attached to these various components and the final solution value is computed. Sensor mobility is given the highest importance while computing the solution values where as connectivity is given the next highest importance followed by functional correlation and coverage provided by the sensor node. The weights that are used for each of these factors are 0.34, 0.25, 0.21, and 0.20. This is due to the reason that in a heterogeneous sensor network each type of sensor has different mobility cost and when moving a mobile node, it should not expend most of its energy in moving from one location to the other. Connectivity provided by the mobile node is given the second highest importance because whatever node has been moved to the failed sensor location, it should be able to relay information to the rest of the sensor network Also functional correlation is given third highest importance because we place importance on the aspect that a failed sensor node has to be replaced by a node that performs at least a subset of tasks that were performed by failed node. Coverage is given least importance because we consider it less important while moving mobile nodes in a medium size heterogeneous sensor network. The solution values that were computed during the simulations performed on the two sensor network models varied from 0.26 to 1.13. The lower level of the solution range corresponds to an unfavorable choice for replacement of a failed static sensor node and the higher limit corresponds to a very good choice for the replacement of a failed static sensor node. In the computation of the solution values, mobility cost is given the highest priority and thus is assigned the highest weight and connectivity is given the next highest weight. One of the reasons for giving mobility cost the highest weight is because of the heterogeneity in the sensor networks.

4.3. Sensor Network Models

For conducting the experiments, two sensor network models were designed with different levels of heterogeneity. The main reason for this choice is to study the behavior of the algorithm under two different sets of environments. Unlike in [2] where the heterogeneous sensor network's life time is assumed to be proportional to the energy of the sensor nodes battery, we consider that in our sensor network models there might be other reasons for the nodes to shut down such as hardware failure, environmental changes and so on. The first sensor network model is an Intrude alert system where the main objective of the system is to detect any intruder in the sensor network area and track it for the most part of the coverage area. The second sensor network model is a sensor network that is used for meteorological purposes which represents a more complex environment than the first sensor network model. In any of the models, we don't consider the tasks that are required by the communication interface of the sensor network. The assumption in designing the sensor network models is that all the different sensor types have some common interface through which they can relay the sensor network information.

4.3.1. Sensor Network Model 1

The first sensor network model is a basic intruder alert system that is made of 3 types of sensors. These 3 types of sensors are required to perform the 4 main functions that are required by the intruder alert system. The 4 main tasks comprise of Temperature measurement, tracking of target, determination of the type of target and Magnetic measurements. These main tasks provide all the necessary functionality required for detection of intruder, tracking and determining the type of intruder. The different target types that can be indentified by this sensor network are metallic objects, non-metallic objects and biological objects. The 3 types of sensors that comprise of the sensor network model are Motion detector, Heat flux sensor and Proximity Sensor. The mobility cost for each of these sensors is determined by the tasks that they are assigned to in the sensor network model. The Motion detector that is considered in this sensor network model is a dual technology motion detector that is a combination of both Pyroelectric infrared sensor and microwave motion detector. These motion detectors provide the functionality of both detecting the intruder at first hand and also gives real time information regarding the position of the target. At first when a Pyroelectric sensor detects an intruder, the microwave motion detector is activated and when it also detects an object, the alarm is activated. In comparison, Pyroelectric sensor draws less power than the microwave radiometer. Also the possibility of false alarms is reduced when using a dual technology motion detector. Heat flux sensor generates a signal depending on the local heat flux and thus can be used in detecting any foreign object in its range. Also, the Heat flux sensor may not provide the intruder exact location as in the case of motion detector sensor but can provide information that the intruder exists with in its range. A proximity sensor is used for detecting the proximity of an object with out any physical contact. These types of sensors usually emit a beam of electromagnetic radiation and looks for changes in the return signal. In this model we considered a combination of photoelectric and inductive sensors.

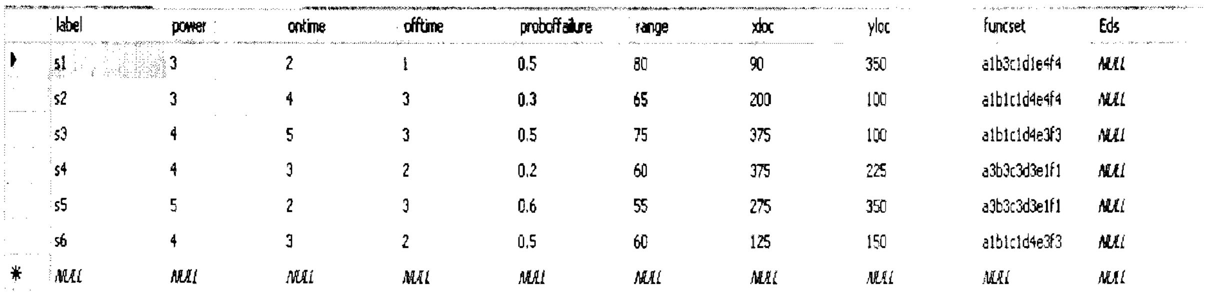
*4.3.2. **Sensor Network Model 2***

The second sensor network model that was designed for performing the experiments is basically for meteorological applications. The sensor types that were chosen for this sensor network are mainly used for measuring meteorological parameters. The size of the second sensor network model is the same as the first sensor network model with only difference being the change in complexity of the environment. In this sensor network model, the number of parameters to be measured by the sensor network is increased to 6 to observe the behavior of the mobile sensor movement algorithm and any changes in the decision making process. Also the number of sensor types is increased to 4 keeping the simulation area or the sensor network area constant. Thus we can say by increasing the number of sensor types and by increasing the number of tasks the sensor network has to perform in a fixed area, we can say that the complexity of the sensor network environment is increased. The four different types of sensors that are used in this sensor network model are Microwave radiometer, Net radiometer, Carbon dioxide sensor and Hygrometer. The mobility cost for each of these sensor types is decided based on the tasks they are assigned to perform in the sensor network. The tasks that the sensor network has to perform are measuring meteorological parameters like measurement of radiation on the earth surface and in the sky, ground surface temperature measurement, air temperature measurement, carbon dioxide measurement, air moisture or water vapor measurement and soil moisture measurement.

All the four different types of sensors together have to accomplish the task of gathering the six different parameters that the sensor network is supposed to measure. Microwave radiometer is a radiometer that measures the energy emitted at microwave wavelengths. Using these measurements, one can calculate a wide variety of atmospheric and surface parameters including soil moisture, sea ice, air temperature, sea surface temperature, salinity, precipitation, the total amount of water vapor and the total amount of liquid water in the atmospheric column directly above or below the instrument. In this sensor network model, microwave radiometer is considered to accomplish the task of measuring moisture, temperature and radiation. A net radiometer is basically used to measure the net radiation at the earth's surface. This is achieved by subtracting the upwelling radiation from the incoming radiation. The type of net radiometer that is used in this sensor network model is a four component design net-radiometer. These types of radiometers are most popular for scientific applications and are used to measure the 4 separate components of the solar radiation that include global solar radiation, reflected solar radiation, infra-red radiation emitted by the sky and infra-red radiation emitted by ground surface. From these four parameters, a different set of parameters can also be calculated. The set of derived parameters include albedo radiation measurement, sky temperature, ground surface temperature and the net radiation measurement. A Carbon dioxide sensor is basically used for the measurement of carbon dioxide gas. The type of sensor that is used in thus sensor network model is the non-dispersive infrared carbon dioxide sensors. These types of sensors are made of four components that include an infra red source, a light tube, an interference filter and an infrared detector. The final result from this sensor would be measurement of the absorption of characteristic wavelength of light. A Carbon dioxide sensor can also be used to measure temperature because of their ability to absorb ambient infrared radiation. Also carbon dioxide sensors can be used to detect fires because fires usually result in an increase in carbon dioxide output.

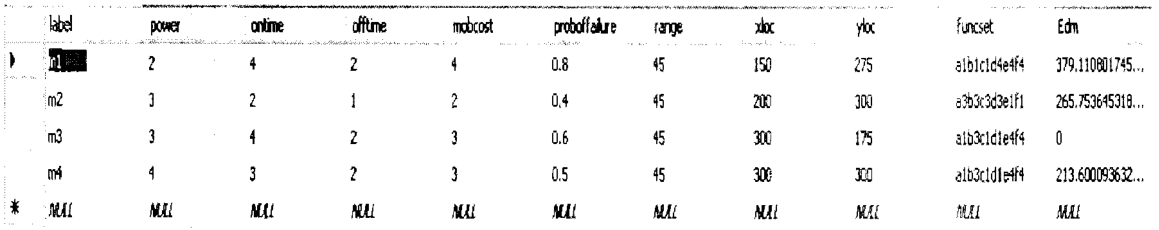
4.4. Sensor Network Data

All the data that is related to the sensor network model is stored in a Sql server database. Different data tables are maintained for storing data regarding the static sensors and mobile sensors. Retrieval of data is done using Active Data objects (ADO) and the same is used for updating the data. For querying the data, stored procedures are used in Sql server for selecting and updating the data about a static or a mobile sensor. The ADO objects and the stored procedures work in co-ordination to ensure the operation performed on the data has been done safely without causing any alteration of other data. The figures in the next page give an idea about the structure of the data that is stored in the Sql server. Separate screenshots have been provided for both the data tables of static sensors data and mobile sensors. The Edm and the Eds fields in the figure below are used to store the Euclidean distance to the failed sensor location.



	label	power	ontime	offtime	probofailure	range	xloc	yloc	funcset	Eds
▶	s1	3	2	1	0.5	80	90	350	a1b3c1d1e4f4	NULL
	s2	3	4	3	0.3	65	200	100	a1b1c1d4e4f4	NULL
	s3	4	5	3	0.5	75	375	100	a1b1c1d4e3f3	NULL
	s4	4	3	2	0.2	60	375	225	a3b3c3d3e1f1	NULL
	s5	5	2	3	0.6	55	275	350	a3b3c3d3e1f1	NULL
	s6	4	3	2	0.5	60	125	150	a1b1c1d4e3f3	NULL
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.3. Static Sensor Datatable.



	label	power	ontime	offtime	mobcost	probofailure	range	xloc	yloc	funcset	Edm
▶	m1	2	4	2	4	0.8	45	150	275	a1b1c1d4e4f4	379.110801745...
	m2	3	2	1	2	0.4	45	200	300	a3b3c3d3e1f1	265.753645318...
	m3	3	4	2	3	0.6	45	300	175	a1b3c1d1e4f4	0
	m4	4	3	2	3	0.5	45	300	300	a1b3c1d1e4f4	213.600093632...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 4.4. Mobile Sensor Datatable.

At the starting phase of the simulation only the x, y and radius of the sensors are retrieved to draw the sensors on the canvas or the simulation area. Later when the sensor failures start to occur, data is updated whenever there has been a change in location of mobile sensor nodes or whenever there is a change in distribution of sensor nodes during the simulation.

4.5. Experimental Results

Experiments were conducted on the two sensor models in the form of a series of sensor node failures. In these experiments, failure of only static sensor nodes is considered and mobile sensor nodes are considered as likely candidates for replacement of failed nodes.

Each simulation consists of a series of static node failures and the number of static node failures is equal to the number of mobile nodes in the sensor network. The series of static node failures for each simulation is called a failure pattern and the algorithm is tested for many failure patterns for each sensor network model. The number of tests conducted for each sensor network model also depends on the number of the types of sensors in the sensor network model.

4.5.1. Experimental Results for Sensor Network Model 1

The experiments conducted in this sensor network model validated many aspects of the algorithm. The number of static sensor nodes in this sensor network model is six and the number of mobile nodes is four. The algorithm was tested for many failure patterns in this sensor network model. Redundant test patterns are removed and the simulation results for six failure patterns are presented. For clarity, each failure pattern is given a simulation code. For example, S1FP1 is an acronym for Simulation for model 1 with Failure Pattern 1. The below table provides the simulation codes for each of the failure patterns and also provides the information about the location of the sensor node failures in each failure pattern.

Failure Pattern Code	**Order of Sensor node failure**
S1FP1	MD(125,150),HF(375,225),PR(90,350),PR(375,100)
S1FP2	MD(125,150),HF(375,225),PR(90,350),MD(275,350)
S1FP3	HF(375,225),MD(125,150),HF(200,100),MD(275,350)
S1FP4	MD(125,150),HF(375,225),MD(275,350),HF(200,100)
S1FP5	HF(375,225),PR(90,350),MD(125,150),PR(375,100)
S1FP6	MD(125,150),PR(90,350),HF(375,225),PR(375,100)

Table 4.1. Simulation codes for Failure Patterns for the first sensor network model

The Simulation codes also provide information on the type of sensor that has failed and its location co-ordinates. Each of the sensor type in this sensor network model is abbreviated in the failure pattern. For example, the failure of motion detector at location (275,350) is mentioned as MD (275,350). The abbreviations for Heat flux Sensor and Proximity Sensor are HF and PR respectively.

Failure pattern	**Initial values**	**S1**	**N1**	**S2**	**N2**	**S3**	**N3**	**S4**	**N4**
S1FP1	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.51, 0.50, 0.70	3. 4th step aborted	0.28, 0.29, 0.67	3.	0.82, 1.06	2. 3rd step aborted	0.55, 0.34, 0.77	3.
S1FP2	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.51, 0.50, 0.70	3. 4th step aborted	0.28, 0.29, 0.7	3.	0.82, 1.06	2. 3rd step aborted	0.48, 0.67, 0.48	3
S1FP3	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.72, 0.94	2. 3rd step aborted	0.70, 0.50	2. 3rd step aborted	0.56, 0.56, 0.96	3.	0.92, 0.73	2. 3rd Step aborted

Table 4.2. Simulation values for the first three failure patterns of sensor model 1

The simulation results for the first three failure patterns are presented in the table above. The computed solution values for each of the sensor node failure are provided and so is the number of steps taken by the algorithm in the simulation to reach the best solution value. The columns with the value S1, S2, S3, S4 denote the solution values computed for each of the first, second, third and fourth sensor node failures in the failure patterns. The number of steps taken in reaching the best solution for each of the sensor node failure is denoted by N1, N2, N3 and N4 .The best solution value returned by the algorithm for each one of the sensor failures is the highest one in the set of solution values. When the first node in a sensor failure pattern fails, the maximum number of steps the algorithm can take under normal temperature scheduling for finding the best suitable sensor is four. This is because under normal annealing schedule, the number of neighborhood candidates at the first sensor failure is always four. It can also be referred to as that all the four mobile sensors are available for evaluation, when the first sensor node fails. After a replacement has been found for the first failed sensor node, the moved mobile sensor node is not available in the next iteration for evaluation. The reason for this elimination is that mobile sensors use much of their power in mobility and powering up at the new location and need time for powering up to be ready to move again. Therefore in a failure pattern, only for the first sensor node failure we have all the four mobile sensors for evaluation.

Later on, there are only three mobile sensors available for next successive sensor failures. In the first sensor failure pattern, when the first static sensor node failed, all the four mobile sensors are available for evaluation. The first two nodes are a bad choice for replacement as they generate solutions that are of very low value. As the annealing schedule is already half way through by the time the first two candidates are generated, the third member generates a very favorable solution value which accelerates the temperature schedule. This in turn causes the annealing schedule to reach its limit and the fourth member is not evaluated. Thus we found a best solution in three steps. But his should not be the true for all cases, as not evaluating the any members after the temperature is accelerated might cause to lose a comparatively more favorable solution in favor of a less favorable solution. Therefore in the current scenario, the algorithm should evaluate at least one candidate after the temperature is accelerated. This in turn also depends at what point in the algorithm, the acceleration in temperature schedule is initiated. If initiated at the early stage of the process the algorithm, in the worst case two candidates in the candidate set would be evaluated in the whole process. This is demonstrated in later failure patterns. In the third failure pattern, when evaluating the candidates for the first sensor node failure, a candidate with a very high solution value is evaluated. At this point, the temperature is accelerated and since this happens at an early stage of the evaluation of candidates, the algorithm proceeds to evaluate the next neighbor in the neighborhood set. This evaluation returns a much more favorable candidate with a higher solution value. At this point, the best solution is updated and the candidate with higher value is returned as the best candidate for replacement of the failed node.

During the failure of sensor nodes, it is sometimes the case that a best solution is not encountered till the last step of the neighborhood evaluation. This is observed in the first failure pattern when the evaluation is started for the second sensor node failure. The first two candidates generate very low solution values (0.28 and 0.29) stating they are least favorable for replacement of the failed sensor node. At this point, during the evaluation of the last sensor node i.e. the third candidate, a high solution (0.68) value is met. This is well above the prescribed best solution value of Smax which is set at 0.45 and the good enough solution value of Smax which is set at 0.55. Also the movement of the initial state is not always initiated because it is decided in a probabilistic way. The probability of moving the initial solution is compared with a random number generator. Due to these reasons, the initial state is not always moved. This is observed in the first failure pattern where in the first sensor node failure, when the first candidate is evaluated, the initial state is moved and best solution is updated and the temperature schedule is accelerated. But in the third sensor node failure, during the evaluation of the second sensor node, best is updated and temperature is accelerated but the initial state is not moved. This in turn validates the design, that the movement of the initial solution is a probabilistic phenomenon and does not depend on the magnitude of the solution value computed. The graph for the solution values in the first three sensor failure patterns is presented below. The notations S_{11}, S_{12}, S_{13}, and S_{14} denote the solution values computed in the first failure pattern for first, second, third and fourth node failures respectively.

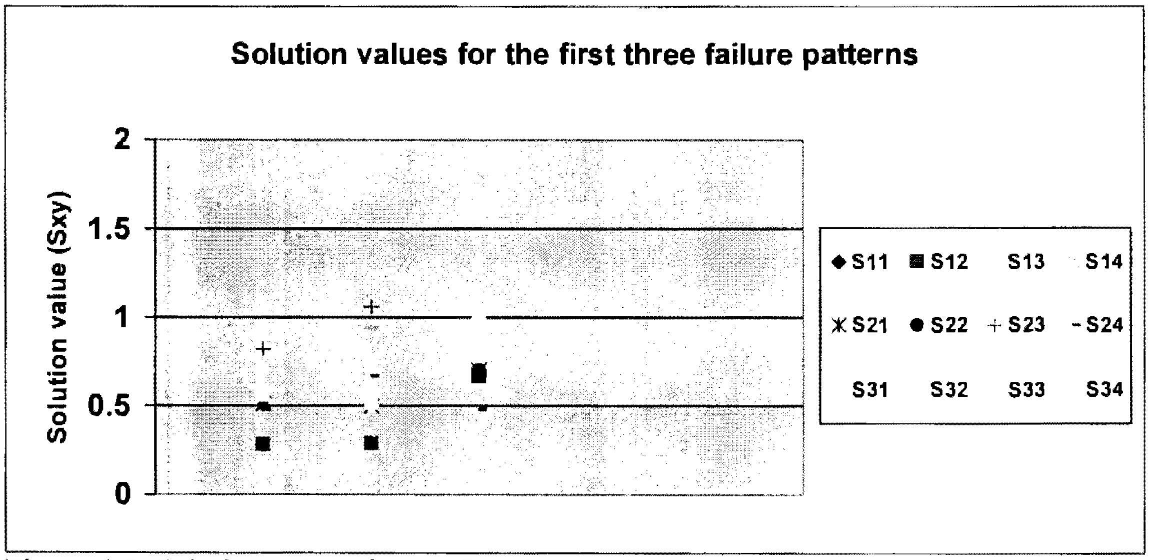


Figure 4.5. Solution values for the first three failure patterns in sensor model 1.

The computation of the Probability of moving the initial solution "Pmov" is computed in a way that it allows uphill movements even in the later parts of the annealing schedule. Thus we can say that moving the initial solution is not a purely probabilistic phenomenon. The simulation results for the next three failure patterns for the first sensor network model are presented in the table below. The table follows the same header notations as the previous table and a graph is also presented pointing out the range of solution values computed in the last three failure patterns. The first three failure patterns in the first sensor network model denote random failures at various points in the network and in various directions. For example failure pattern 2 traverses the sensor nodes in a loop in the interior of the sensor network i.e. the failure pattern returns to a point close to location of the previous sensor node failure. The last two sensor node failures traverse the boundaries of the sensor network in different directions while the fourth sensor failure pattern starts at a boundary and traverses the interior of the sensor network in a loop.

Failure pattern	**Initial values**	**S1**	**N1**	**S2**	**N2**	**S3**	**N3**	**S4**	**N4**
S1FP4	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.51, 0.50, 0.70	3. 4th step aborted	0.28, 0.29, 0.7	3	0.51, 0.73, 0.92	3.	0.31, 0.31, 0.71	3.
S1FP5	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.72, 0.94	2. 3rd step aborted	0.82, 1.06	2. 3rd step aborted	0.51, 0.70, 0.69	3	0.55, 0.34, 0.77	3
S1FP6	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.51, 0.50, 0.70	3	0.82, 1.06	2. 3rd step aborted	0.28, 0.70, 0.69	3	0.36, 0.54, 0.77	3

Table 4.3. Simulation values for the next three failure patterns of sensor model 1

During the evaluation of the first sensor node in the fourth failure pattern the best state and the initial solution were updated but temperature was not accelerated because the computed solution was not good enough for acceleration to be set. This validated another set of possible state moves by the algorithm. Another proof was encountered showing that the movement of the initial solution is independent on the magnitude of the solution computed. During the evaluation of candidates during the second sensor node failure, the first two candidates generated very low solution values and only the initial state was moved. This was also observed during the evaluation of the first candidate during the failure of fourth sensor node in the last failure pattern. Also it has been observed that the chance of finding a good solution at the early stages of the algorithm depends on the distribution of the sensor nodes too. The direction in which the failure patterns traverse also determine at what step the best solution is computed in the algorithm. In the fifth failure pattern, it is noted that the temperature has to be accelerated at least at the second step in order for the algorithm to calculate at least one candidate solution after the annealing schedule is accelerated. Therefore we can set a limit that in order for at least one candidate to be evaluated after the temperature is accelerated, the temperature acceleration has to occur no later than the second step else the algorithm would come to a halt and would return the best state and solution it evaluated. The graph for the solution values for the last three failure patterns is presented in the next page.

The re-arrangement of the sensor nodes due to the series of sensor node failures resulted in reduced coverage in the overall sensor network and also creates entry points at the boundaries of the sensor network which resulted in breach paths in the sensor network. A breach path starts at a point on the boundary of the sensor network and traverses through certain part of the sensor network or it can also be a path that starts at one point on the boundary and ends at the any other end of the sensor network. If two adjacent sensor node locations that were once active have failed and their locations were not filled by any other mobile sensor node then we take that as one entry point for a breach path.

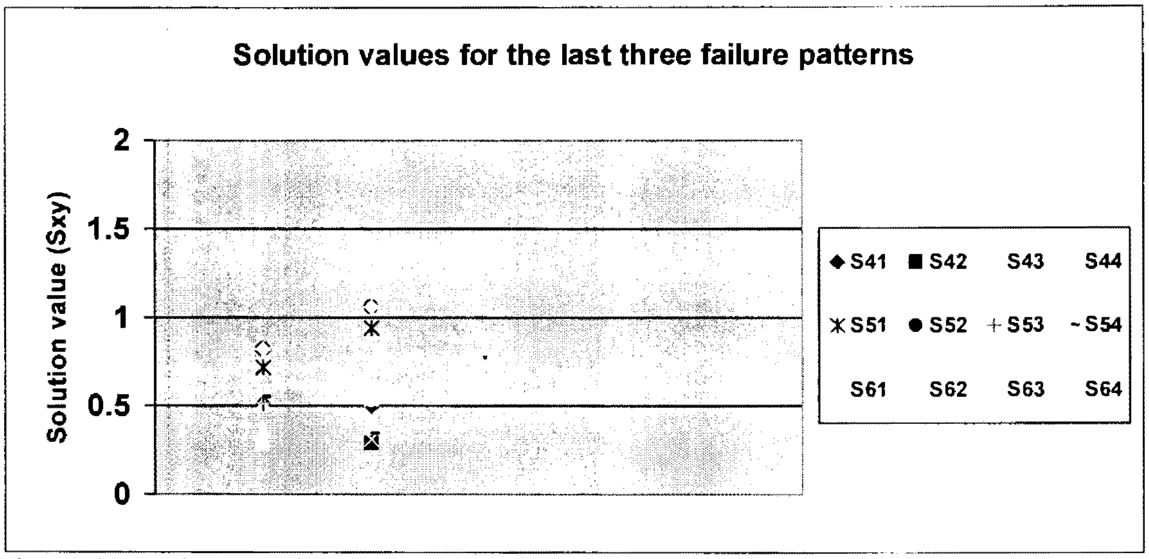


Figure 4.6. Solution values for the last three failure patterns in sensor model 1.

So the number of breach paths in a sensor network will depend on the number of ways an intruder can traverse through any one of the entry points. In the simulations conducted in the first sensor network model, the number of entry points for each of the failure pattern is presented in the table below. Since this sensor network model is designed to be an intruder alert system, these entry points can be seriously degrade the objective of deploying this system.

Failure Pattern Code	**Number of entry points**
S1FP1	3
S1FP2	4
S1FP3	3
S1FP4	3
S1FP5	4
S1FP6	4

Table 4.4. Number of entry points in sensor network for sensor model 1

4.5.2. __Experimental Results for Sensor Network Model 2__

The second sensor network model is designed to have to a high degree of heterogeneity when compared to the first model and also has more type of sensors. The experiments conducted consisted of failure patterns that are made in a similar way as in the first sensor model. The number of sensor node failures in each failure pattern is equal to the number of mobile nodes in the sensor network model. In this sensor network model, the number of static nodes is six and the number of mobile nodes is four. The number of types of sensors nodes is four. The simulation data for this sensor network model is presented in a similar way as the previous sensor network model. For clarity of the data, each of the simulation or the failure pattern is assigned a simulation code in a similar way as the previous model. The first four failure patterns for the second sensor network model are presented in the below table.

Failure Pattern Code	**Order of Sensor node failure**
S2FP1	NR(200,100),MR(125,150),CO2(90,350),MR(375,100)
S2FP2	MR(125,150),CO2(90,350),HR(375,225),MR(375,100)
S2FP3	MR(375,100),NR(200,100),MR(125,150),CO2(90,350)
S2FP4	NR(200,100),MR(125,150),MR(375,100),MR(275,350)

Table 4.5. Simulation codes for the first four failure patterns in sensor network model 2

Like in the simulations of the first sensor network model, the four sensor types in the sensor network model are denoted with their abbreviations in the failure pattern. Failure of microwave radiometer at location (125,150) is denoted as MR (125,150). The abbreviations for Net radiometer, Hygrometer and Carbon dioxide sensors are NR, HR and CO2 respectively. In the above failure patterns, the first two start at the same point at one end of the sensor network and end at different locations. The third failure pattern starts at one end of sensor network and follow a random path of failure and end at an opposite locations. Te fourth failure pattern is similar to the first two failure patterns in that it follows a linear path in the sensor node failures. The simulation data for the first four failure patterns is presented in the table below. The table headings S1, S2, S3 and S4 denote the solution values computed for the first, second, third and fourth sensor node failures in a failure pattern and the N1, N2, N3 and N4 denote the number of steps taken in the algorithm for reaching the best solution value.

Failure pattern	**Initial values**	**S1**	**N1**	**S2**	**N2**	**S3**	**N3**	**S4**	**N4**
S2FP1	Sbest =0.45, Smax =0.55, Tmax =6, dT=2	0.73, 0.72	2. 3rd step aborted	0.95, 0.54	2. 3rd step aborted	0.60, 0.78	2. 3rd step aborted	0.77, 0.76	2. 3rd step aborted
S2FP2	Sbest =0.45, Smax =0.55, Tmax =6, dT=2	0.70, 0.29	2. 3rd step aborted	0.60, 0.78	2. 3rd step aborted	0.96, 0.75	2. 3rd step aborted	0.78, 0.37	2. 3rd step aborted
S2FP3	Sbest =0.45, Smax =0.55, Tmax =6, dT=2	0.78, 0.77	2. 3rd step aborted	0.72, 0.31	2. 3rd step aborted	0.95, 0.96	2. 3rd step aborted	0.60, 0.79	2. 3rd step aborted
S2FP4	Sbest =0.45, Smax =0.55, Tmax =6, dT=2	0.73, 0.72	2. 3rd step aborted	0.95, 0.54	2. 3rd step aborted	0.77, 0.76	2. 3rd step aborted	0.76, 0.75	2. 3rd step aborted

Table 4.6. Simulation values for the first four failure patterns in sensor network model 2

In the first sensor failure pattern, all the node evaluations at the first step were favorable solution values. Due to this the annealing schedule was in acceleration mode starting from the first evaluation. The evaluation of the candidates for the failed sensors was completed in the first two steps. The best solutions achieved were encountered either in the first or the second step and were much better than the good enough solution Smax value. The same evaluations were observed during the second failure pattern. All the evaluations of the failed sensor nodes were completed in two steps and the best solutions were better than the good enough solution. Both these failure patterns started at some point outside the sensor network and ended at a point on the opposite side of the sensor network. Due to the random pattern of the third failure pattern, the evaluation of the candidates of the failed nodes was completed in two steps. Some of the solution values encountered in the evaluation process were far better than the first two failure patterns. The results for the fourth failure pattern are similar to the first two failure patterns and the best solution values were slightly better than the first two evaluation patterns. The graph for the solution values for the first four failure patterns is presented is presented below.

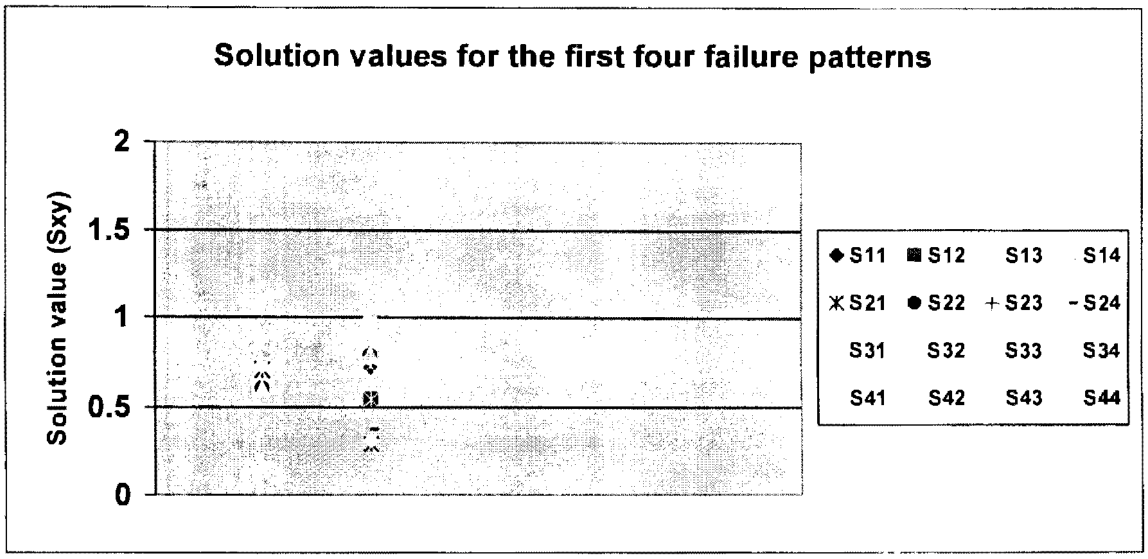


Figure 4.7. Solution values for the first four failure patterns in sensor model 2.

The notations used to represent the solution values are similar to the ones used in the previous graphs i.e. the notations S_{11}, S_{12}, S_{13}, S_{14} and S_{15} denote the solution values computed in the first failure pattern for first, second, third, fourth and the fifth node failures respectively. Unlike in the previous sensor model, it can be seen that none of the solution values exceeded the value of 1.The simulation codes for the next four failure patterns are given below in the table.

Failure Pattern Code	**Order of Sensor node failure**
S2FP5	NR(200,100),MR(375,100),HR(275,350),CO2(90,350)
S2FP6	HR(375,225),MR(125,150),CO2(90,350),MR(375,100)
S2FP7	CO2(90,350),HR(375,225),HR(275,350),NR(200,100)
S2FP8	CO2(90,350),HR(375,225),MR(125,150),HR(275,350)

Table 4.7. Simulation codes for the last four failure patterns in sensor network model 2

The fifth failure pattern starts at same end as the first failure pattern and follows a linear path in the sensor network. The sixth, seventh and eight failure patterns follow a random path as in the case of the third failure pattern but traverse the network in different directions. Also the starting points of both the third and sixth failure patterns are same but the directions are different. These random failure paths can be considered to be mirror images of each other with the failure patterns traversing the network in different directions. The simulation data for these failure patterns is presented in the next page in a table. The table headers follow the same notations as used before in the previous simulation data tables. The number of steps field also says at what point in the algorithm the search for the best solution was aborted due to the end of annealing schedule.

Failure pattern	Initial values	S1	N1	S2	N2	S3	N3	S4	N4
S2FP5	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.73, 0.72	2. 3rd step aborted	0.77, 0.76	2. 3rd step aborted	0.76, 0.97	2. 3rd step aborted	0.80, 0.84	2. 3rd step aborted
S2FP6	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.51, 0.75, 0.96	3. 4th step aborted	0.71, 0.70	2. 3rd step aborted	0.80, 1.04	2. 3rd step aborted	0.37, 0.77, 0.76	3.
S2FP7	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.80, 0.85	2. 3rd step aborted	0.52, 0.75, 0.51	3.	0.47, 0.26, 0.69	3.	0.72, 0.73	2. 3rd step aborted
S2FP8	Sbest= 0.45, Smax =0.55, Tmax =6, dT=2	0.80, 0.85	2. 3rd step aborted	0.52, 0.75, 0.51	3.	0.70, 0.71	2. 3rd step aborted	0.47, 0.69, 0.47	3.

Table 4.8. Simulation values for the last four failure patterns in sensor network model 2

Since the fifth failure pattern starts at one end of the sensor network and follows a linear path like in the case of the first failure pattern, the algorithm behaves in a similar way as it was during the evaluation of candidates for the sensor node failure. The solution values that were computed in the fifth failure pattern were much closer to each other unlike in the case of the first failure pattern where there was significant difference between some of the solution values computed for each sensor node failure. In the fifth failure pattern, most of the solution values were closely packed. Also the number of steps taken for reaching the best solution in each of the sensor node failure is two.

The sixth, seventh and the eight sensor failure patterns are similar to the third sensor failure pattern in that they are random in nature and so the algorithm behaves in a similar way as it did during the third failure pattern. But the maximum number of steps taken to reach the best solution is three. This could be attributed to the fact that even though the sensor failure pattern was random, so was also the picking of the neighborhood sensors for evaluation. In the sixth, seventh and the eight sensor failure patterns, half the best solutions were reached in two steps in the algorithm while the rest of the best solutions took three steps in the algorithm. In the last three failure patterns the occurrence of the best solution evaluations that took three steps in the algorithm were independent of the number of node failure in the test pattern. Thus we can say that in a medium size heterogeneous sensor network when random failure of sensor nodes occurs, the best solutions can be found in a maximum of three steps. The graph for the solution values for the last four failure patterns in presented in the chart below.

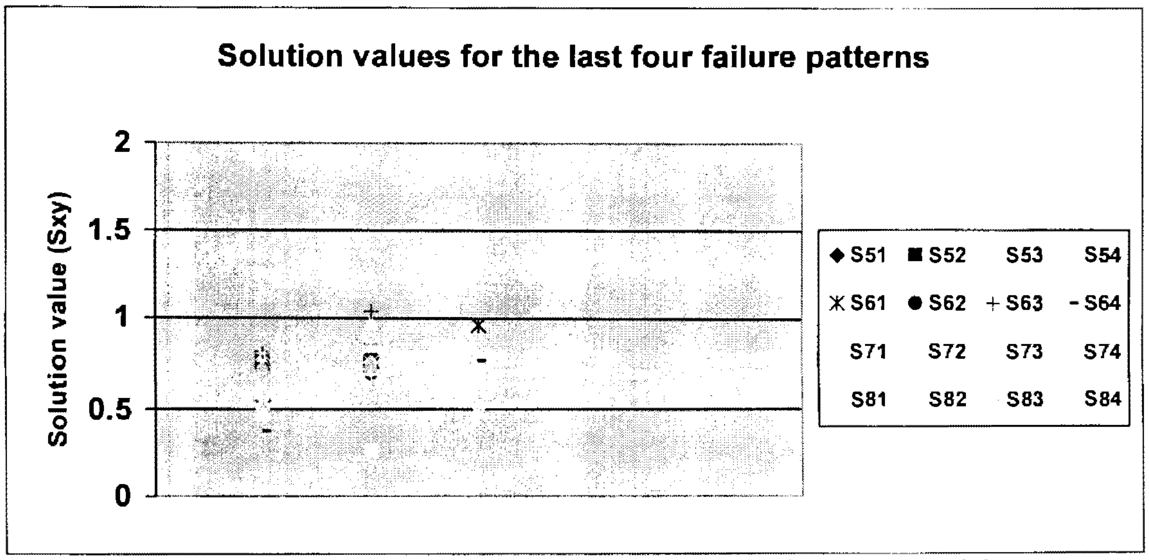


Figure 4.8. Solutions values for the last four failure patterns in sensor model 2.

Like in the first sensor network model, there was considerable reduction in the coverage area of the sensor network due to replacing a large static node with a mobile node of lesser range. Since in the second sensor network model, the main objective is to measure the atmospheric parameters, the concept of breach paths might be a little absurd but can affect the overall objective of the deployment of the sensor network. In the simulations, the number of entry points for a breach path varied based on the failure pattern. This can be attributed to the increase in heterogeneity of the sensor network. Since there are more types of sensors nodes in the sensor network, it appears to be that when a sensor node failure occurs, the network has greater options than in the case of the previous sensor model. The number of entry points for a breach path in each of the simulations conducted is presented in the table below.

Failure Pattern Code	**Number of entry points**
S2FP1	1
S2FP2	3
S2FP3	3
S2FP4	3
S2FP5	2
S2FP6	4
S2FP7	3
S2FP8	1

Table 4.9. Number of entry points in the sensor network in sensor network model 2

Unlike in the previous sensor model, it can be observed that the number of entry points for the breach paths in the sensor network varied according to the failure pattern in the sensor network and the values were continuous over a fixed range starting from a minimum value of one to a maximum entry point value of four. If there was one entry point then it is because that the sensor network has no coverage or very little coverage on one side of the sensor network and having four entry points means that there is a possible breach path for each side of the sensor network. Thus we can infer that as the heterogeneity or the number of sensor types in a sensor network increases, the number of entry points for a breach path changes according to the failure pattern. While in most cases in the previous sensor model, the number of entry points was near the maximum value, in the second model it can be said that there was just one scenario were there were entry points from all sides of the sensor network and in most case there were three entry points and in the best case scenario it was just one entry point. Though the second model showed better results in the number of entry points, the coverage area was the same as the first model.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1. Conclusions

The results obtained from the test failure patterns in the two sensor network models pointed out many factors that have to be taken into consideration when moving mobile sensor nodes in a heterogeneous sensor network. Although the main factors that were considered when designing the mobile sensor movement algorithm yielded the desired results, it pointed out a few other factors that will be useful in getting better results. The evaluation of the candidates for the replacement of the failed nodes worked as per the design of the algorithm. Though this algorithm is designed for restarting itself in the event when no favorable solution has been found at the end of annealing schedule, it was not observed to happen in any of the test patterns in the two sensor network models. Since we used two sensor network models for evaluating the performance of out algorithm, we present their conclusions separately.

5.2. Conclusions for Sensor Network Model 1

The first sensor network model represents a heterogeneous environment of medium level complexity. It has just three types of sensor nodes and the commonality of the task performed by the sensors ranges from very low to high among the 3 sensor types. Thus we can refer to it as a medium level complex environment. The main purpose of testing the algorithm for this environment is to validate the various working aspects of the algorithm for a sensor network of medium level heterogeneity. The following are the conclusions of the experiments performed for this sensor network model:

1. The mobile sensor movement algorithm performed according to the design aspects in this sensor network model.
2. Since the first sensor network model is a medium sized sensor network, the algorithm performed well for various failure patterns and there was no instance where it failed to find a suitable replacement node. So we can say that algorithm was scalable to medium sized sensor networks.
3. The algorithm was found to perform well for all the test failure patterns. Hence we can say that the algorithm performed well for medium level complex environments.

The annealing schedule made the algorithm skip the neighborhood candidate evaluations whenever a good enough solution was found and solution values were in accordance to sensor parameters. The solution values were low when the sensor node being evaluated was not suitable for replacement of a failed node and gave high values for sensor nodes that were suitable candidates for replacement of the failed node. The failure of a sensor node of particular type had to be replaced by a sensor node of the same type or a node that does at least some of the failed node functions while consuming reasonable amount of sensor network resources. The node that performs the highest number of failed node tasks while maintaining the connectivity of the network with reasonable mobility cost is preferred as replacement.

Though the movement of the sensor nodes is performed accordingly, there were a few instances where the replacement of the sensor nodes resulted in loss of connectivity to a particular area in the sensor network. This needs to be corrected by taking measures that will result in readjustment of the sensor nodes after a mobile node has been moved to the failed sensor location. Also since this model is based on detection of foreign objects the presence of breach paths or the entry points for an intruder into the network resulted in compromising the objective in deploying the sensor network. Also the coverage of the senor network is reduced in most of the test failure patterns.

5.3. <u>Conclusions for Sensor Network Model 2</u>

The second sensor network model represents a high level of heterogeneity in the sensor network. More number of tasks was performed by nodes in the second sensor network model and more types of sensor nodes were taken when compared to the first sensor network model. The commonality of the tasks performed by various types of nodes varied from low to high among the different sensor types. The main purpose of testing the algorithm for this sensor network model is to validate the working of the algorithm in a high heterogeneous environment and observe its working for various failure patterns. The conclusions for various failure patterns in second sensor network model are presented below:

1.) The mobile sensor movement algorithm performed well according to the design aspects in the second sensor network model even thought it resulted in a few extra computations when compared to the first sensor network model.

2.) The algorithm performed well in all the failure patterns and there was no instance where algorithm failed to find a suitable node for replacing a failed sensor node.

3.) Thus we can say that the algorithm is scalable for medium sized sensor networks.

Even though the second sensor network model is a more distributed than the first model, there was no instance where the algorithm failed to find a suitable node for replacement. Hence we can say that the algorithm works well for a medium sized complex network

The failure patterns that were implemented in the sensor network model resulted in movement of the suitable sensor nodes to the failed sensor location. Due to higher level of heterogeneity in the sensor network which might be attributed to more number of sensor types in the sensor network and due to more task commonality, the failure of a sensor node at a particular location resulted in computation of more number of favorable solution values during the algorithm process. This in turn led to placement of sensor nodes with out sacrificing the tasking required at the failed sensor node location. Also a favorable node yielded a higher solution value where as less favorable node yielded a low value.

Also as in the case of the first sensor network model, in some test patterns, loss of connectivity was observed in some part of the sensor network which will need some corrective measures like readjustment of sensor nodes after moving a mobile node to a failed sensor node location. Also the less number of entry points were observed in some failure patterns in the sensor network for the breach paths to develop. This can be because of the high task sharing in the sensor network so if one sensor node of a particular type fails, then it does not have to be replaced by the sensor node of the same type. The number of sensors that partially perform the failed nodes functions are more than in the previous case. Limitations of the algorithm on the complexity and scalability factors are discussed in a more detailed way in the next section.

5.4. Complexity and Scalability of the Algorithm

In this section we discuss in more detail about the scalability and the complexity factors of the algorithm and present its limitations with respect to these two factors.

5.4.1. Complexity of the Algorithm

The algorithm was found to function properly in a medium sized network where at least majority of the sensor types shared at least one third of the tasks in common. The following are the limitations of the algorithm with respect to complexity of the sensor network environment:

1.) The algorithm is found to restart the search process before finding the best suitable node in environments that are more complex i.e. where sensor types share less similarity in the tasks they perform and have different characteristics such as range, mobility cost and so on.

2.) As the complexity of the network increases in a medium sized network the algorithm takes more computation time (by restarting itself) in finding the best solution and when the complexity reaches a point where a failed node is surrounded by many unsuitable candidates , the algorithm is found to fail.

3.) Thus the algorithm is found to handle sensor node failures in complex medium sized sensor network environments such as the two sensor network models that were presented.

5.4.2. Scalability of the Algorithm

Below are the conclusions regarding the scalability factor of the algorithm:

1.) The algorithm was found to handle sensor node failures in a medium sized network

2.) It is found that the failure rate of algorithm increases as the size of the sensor network increases from the proposed sensor network models.

3.) As the size of the sensor network increases from the size of the two sensor network models, it is found that the algorithm fails more frequently. Also in a large sized network, it is observed that the failure of the algorithm also depends on the failure pattern of the sensor nodes. A failure pattern can trigger an unfavorable distribution of mobile sensors for a particular failed sensor.

4.) Also the algorithm is more bound to restarts as the size of the network grows and is found to fail when the complexity of the network increases.

5.) One more instance where the algorithm is found to fail is in a large sensor network, when a failed node is surrounded by sensor types of dissimilar type i.e. when the neighborhood of the failed node mostly comprises of unfavorable candidates.

5.5. Future Work

The algorithm proposed in this paper yields the good results for a medium sized heterogeneous sensor network. As mentioned before, the loss in connectivity in some of the failure patterns has to be taken care by readjustment of the nodes after a failed node has been replaced by a mobile node. The same approach can be taken to reduce the number of breach paths in the sensor network. Improvements in the algorithm will include handing large sensor networks where there are many types of sensor nodes and these nodes in the network might have different levels of task sharing. The future improvements will include handling of the failure of the sensor nodes in a distributed environment. One of the challenges in a distributed environment includes competition for the network resources. Algorithms have been proposed [15] for the deployment of the sensor nodes based on the resources the sensor nodes possess. In distributed environment, a single annealing schedule will not be good enough for finding the suitable solution for the failed sensor node. It might require multiple annealing schedules or parallel searches that search the entire sensor network simultaneously. One of the improvements to the basic Simulated Annealing procedure is the adaptive simulated annealing technique. Also if a sensor has failed in a sensor network it does not has to be replacement by a node from the same network. To counter for multiple failures a group of nodes can be moved from one sensor network to other sensor network. One such method was proposed in [5] which proposes movement of a group of sensors between two heterogeneous sensor networks. Also parallelization of the basic simulated annealing technique as proposed in [14] can be used for countering simultaneous node failures. In a complex environment such as in a large sensor network made of many different types of sensor nodes, the main challenge would be not to encounter a local minima problem. For this one of the solutions could be to launch a many search processors simultaneously in the sensor network. An another alternative would be that if a search process terminates at one location in a sensor network with no solution being found, then the search could start at some other random point in the network with a different set of initial parameter values. To counter the loss in connectivity in some of the failure patterns, algorithms such as [8] can be implemented after the nodes have been moved to failed sensor locations. Also to counter the breach paths in a medium sized network where there is less or no task sharing among sensor nodes, additional mobile nodes needs to be deployed in the sleep mode. Thus one of the improvements that the algorithm might incorporate is to switch off a sensor node if it is found that it is not required at that particular location and is found using up the network resources. This feature can also be referred to as energy management in a heterogeneous sensor network and will require an analysis of the energy consumption and lifetime factors of the sensor network. In our future simulations that might involve large sensor networks with many different types of sensor nodes , an approach similar to that as proposed in [3] might be taken to make our simulations involve more parameters that were not considered in the initial simulations.

REFERENCES

[1] Enrique J. Duarte-Melo, Mingyan Liu. "Analysis of Energy Consumption and Lifetime of Heterogeneous Wireless Sensor Networks" www.eecs.umich.edu/~mingyan/pub/sensor-globecom02.pdf "Global Telecommunications Conference", 2002, November 2002.

[2] Vivek Mhatre, Catherine Rosenberg, Daniel Kofman, Ravi Mazumdar and Ness Shroff. "A Minimum Cost Heterogeneous Sensor Network with a Lifetime Constraint" http://www.ece.uwaterloo.ca/~cath/tmc04.pdf "IEEE Transactions on Mobile Computing", Jan-Feb 2005.

[3] Lewis Girod, Thanos Stathopoulos, Nithya Ramanathan, Jeremy Elson, Eric Osterweil, Tom Schoellhammer, Deborah Estrin. "A System for Simulation, Emulation and Deployment of Heterogeneous Sensor Networks" http://lecs.cs.ucla.edu/~girod/papers/emtos-sensys04.pdf "Proceedings of the 2nd International conference on Embedded networked sensor systems", 2004.

[4] Rajesh Krishnan, Carla C. Purdy. "Comparison of Simulated Annealing and Genetic Algorithm approaches in optimizing the output of Biological Pathways" www.ece.uc.edu/~cpurdy/preprints/ANNIE_06.pdf "Proceedings of ANNIE Conference", 2006.

[5] Patrick Traynor, JaeSheung Shin, Bharat Madan, Shashi Phoha. "Efficient Group Mobility for Heterogeneous Sensor Networks" http://nsrc.cse.psu.edu/tech_report/NAS-TR-0026-2005.pdf "Vehicular Technology Conference, 2006", 25-28 September 2006.

[6] William L. Goffe, Gary D. Ferrier, John Rogers. "Global Optimization of Statistical Functions with Simulated annealing" http://goffe.oswego.edu/anneal-preprint.pdf "Journal of Econometrics" Volume: 60, Issue: 1-2 (), Pages: 65-99, 1994.

[7] Vivek Mhatre, Catherine Rosenberg. "Homogeneous vs. Heterogeneous Clustered Sensor Networks: A Comparative Study" www.ece.uwaterloo.ca/~cath/icc04.pdf "IEEE International Conference on Communications, 2004", pages: 3646- 3651, Volume: 6, 20-24 June 2004.

[8] Kenneth A. Berman, Fred S. Annexstein, Aravind Ranganathan. "Dominating Connectivity and Reliability of Heterogeneous Sensor Networks" http://fawn2006.conf.citi.insa-lyon.fr/Slides/Berman_FAWN2006.pdf "Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, 2006", 13-17 March 2006.

[9] Akis Spyropoulos, Cauligi S. Raghavendra and Viktor K. Prasanna. "A Distributed Algorithm for Waking-up in Heterogeneous Sensor Networks" http://www.springerlink.com/content/qmmct1xhg5p00bt5/ "Information processing in sensor networks", 22-23 April 2003.

[10] Yun Wang, Xiaodong, Wang, Dharma, P. Agrawal and Ali A. Minai. "Impact of Heterogeneity on Coverage and Broadcast Reachability in Wireless Sensor" Networks", www.ece.uc.edu/~aminai/papers/wang_ICCCN06.pdf "15th International Conference on Computer Communications and Networks", 2006, Pages 63-66, 9-11 Oct. 2006.

[11] Xiaojiang Du, Fengjing Lin. "Maintaining Differentiated Coverage in Heterogeneous Sensor Networks",

www.hindawi.com/journals/wcn/2005/269210.pdf "EURASIP Journal on Wireless Communications and Networking", Volume 2005, Issue 4, Pages: 565 - 572, September 2005.

[12] Jan Beutel." Metrics for Sensor Network Platforms", www.tik.ee.ethz.ch/~beutel/pub/B2006a.pdf, "Proceedings of ACM Workshop on Real-World Wireless Sensor Networks", ACM Press, New York, pages 26-30,June 2006.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., M. P. Vecchi. "Optimization by Simulated Annealing" www.eecg.utoronto.ca/~janders/ece1387/readings/sim_anneal.pdf "Journal of Computer Science", Volume: 220 Pages: 671-680, May 1983.

[14] Jonas Knopman, Julio S. Aude. "Parallel Simulated Annealing: an Adaptive Approach" http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=580950, "11th International Proceedings of Parallel Processing Symposium", 1997, Pages 522-526, 1-5 April 1997.

[15] Yong Ma, Siddharth Dalal, Majd Alwan, James Aylor. "ROP: A Resource Oriented Protocol for Heterogeneous Sensor Networks" www.agingtech.org/documents/ROP_full.pdf "Virginia Tech Symposium on Wireless Personal Communications", Pages 59-70, June 2003.

[16] Andrea Lecchini-Visintini, John Lygeros, Jan Maciejowski. "Simulated Annealing: Rigorous finite-time guarantees for optimization on continuous domains" http://books.nips.cc/papers/files/nips20/NIPS2007_0063.pdf "Neural Information Processing System (NIPS07)", December 3-6 2007.

[17] Franco Busetti. “Simulated annealing overview”
www.geocities.com/francorbusetti/saweb.pdf , Report , 2003.

[18] Lin Lin Chen, Stephen M. Pollock. “Yield Optimization by Simulated Annealing”
http://ioe.engin.umich.edu/techrprt/pdf/TR91-23.pdf “Engineering Design and Automation Journal”, Aug 1996.

[19] Mark Yarvis, Nandakishore Kushalnagar, Harkirat Singh, Anand Rangarajan, York Liu, Suresh Singh. “Exploiting Heterogeneity in Sensor Networks”
http://web.cecs.pdx.edu/~singh/ftp/singh05infocom.pdf “INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE Proceedings” Volume 2, Pages: 878- 890, 13-17 March 2005.

[20] Bradley J. Buckham (93-07482), Casey Lambert (99-05232). “Simulated Annealing Applications”
http://www.me.uvic.ca/~zdong/courses/mech620/SA_App.PDF , 18 November 1999.

[21] Granville, V.Krivanek, M.Rasson, “Simulated annealing: a proof of convergence”
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=295910 “IEEE Transactions on pattern analysis and machine intelligence”, volume 16, issue 6, pages 652-656, Jun 1994.