3-26-2013

Wireless Sensor Network Deployment

Yipeng Qu
Florida International University, yqu002@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

WIRELESS SENSOR NETWORK DEPLOYMENT

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL ENGINEERING

by

Yipeng Qu

2013

To: Dean Amir Mirmiran
College of Engineering and Computing

This dissertation, written by Yipeng Qu, and entitled Wireless Sensor Network Deployment, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Jean H. Andrian

Kang K. Yen

Mohammed Hadi

Manos M. Tentzeris

Stavros V. Georgakopoulos, Major Professor

Date of Defense: March 26, 2013

The dissertation of Yipeng Qu is approved.

Dean Amir Mirmiran
College of Engineering and Computing

Dean Lakshmi N. Reddi
University Graduate School

Florida International University, 2013

DEDICATION

I dedicate this dissertation to my family and my loving boyfriend. Without their love, patience, understanding and support, the completion of this work would never have been possible.

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advising professor, Dr. Stavros V. Georgakopoulos, who has continuously given me guidance and excellent support throughout my entire Graduate and Doctoral studies at FIU.

I would like to thank my committee members, Professor Jean H. Andrian, Professor Kang K. Yen, Professor Manos M. Tentzeris and Professor Mohammed Hadi for serving on my dissertation defense committee and giving enlightening comments.

I am extremely grateful to my loving family in China and my darling boyfriend and proofreader Andrés Miño, who have supported me throughout my research.

Finally, I would like to sincerely thank all of my fellow members of the Electrical and Computing Engineering department and specially all members in our research group: Olutola Jonah, Hao Hu, and Shun Yao for their help and friendship.

ABSTRACT OF THE DISSERTATION

WIRELESS SENSOR NETWORK DEPLOYMENT

By

Yipeng Qu

Florida International University, 2013

Miami, Florida

Professor Stavros. V. Georgakopoulos, Major Professor

Wireless Sensor Networks (WSNs) are widely used for various civilian and military applications, and thus have attracted significant interest in recent years. This work investigates the important problem of optimal deployment of WSNs in terms of coverage and energy consumption. Five deployment algorithms are developed for maximal sensing range and minimal energy consumption in order to provide optimal sensing coverage and maximum lifetime. Also, all developed algorithms include self-healing capabilities in order to restore the operation of WSNs after a number of nodes have become inoperative.

Two centralized optimization algorithms are developed, one based on Genetic Algorithms (GAs) and one based on Particle Swarm Optimization (PSO). Both optimization algorithms use powerful central nodes to calculate and obtain the global optimum outcomes. The GA is used to determine the optimal tradeoff between network coverage and overall distance travelled by fixed range sensors. The PSO algorithm is used to ensure 100% network coverage and minimize the energy consumed by mobile and range-adjustable sensors. Up to 30% - 90% energy savings can be provided in different scenarios by using the developed optimization algorithms thereby extending the lifetime of the sensor by 1.4 to 10 times.

Three distributed optimization algorithms are also developed to relocate the sensors and optimize the coverage of networks with more stringent design and cost constraints. Each algorithm is cooperatively executed by all sensors to achieve better coverage. Two of our algorithms use the relative positions between sensors to optimize the coverage and energy savings. They provide 20% to 25% more energy savings than existing solutions. Our third algorithm is developed for networks without self-localization capabilities and supports the optimal deployment of such networks without requiring the use of expensive geolocation hardware or energy consuming localization algorithms. This is important for indoor monitoring applications since current localization algorithms cannot provide good accuracy for sensor relocation algorithms in such indoor environments. Also, no sensor redeployment algorithms, which can operate without self-localization systems, developed before our work.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Problem statement

Wireless sensor networks have recently attracted significant interest in the engineering community and among academic researchers. Different sensor network application testbeds have been built by various universities, such as the habitat monitoring sensor networks by the University of California at Berkeley and the College of Atlantic [1], the Zebranet Project for monitoring animal habits by Princeton University [2], and the wireless sensor network monitoring volcano activities in Ecuador by Harvard University, the University of New Hampshire, and the University of North Carolina together [3]. Also, DARPA has built a self-healing, smart minefield based on wireless sensor networks [4].

Different research areas are related to wireless sensor networks, such as sensor network security, coverage, communication, etc. Specifically, this dissertation focuses on the area coverage problem of mobile and wireless sensors. In the area coverage problem, each sensor covers a particular sub-area, and the total covered area of the sensor network is made up of the individual covered areas of each sensor node. Maximizing the total coverage area of the entire sensor network is the major objective of the area coverage problem. The area coverage problem is closely related to the performance of systems in applications, such as, target detection and tracking, monitoring the battlefield, homeland security, personal protection, and animal habit monitoring.

The location of the sensors is the most critical factor related to the network coverage. In order to obtain a good coverage, sensors are deployed deterministically.

However, there are certain conditions when sensors cannot be deployed deterministically, for example in applications involving areas of natural disasters or harsh environments. Additionally, if the sensing field is very large or has only limited entrance, sensors may not be able to be deployed one by one in precise locations. They may instead be deployed all at once from an aircraft or similar vehicles.

When sensors are randomly deployed, the area-coverage initially provided by the sensor network cannot be guaranteed to be optimal, as in the deterministic deployment. An example of a randomly deployed sensor networks is shown in Figure 1.1. There are twelve sensors deployed in a rectangular sensing field. In the left part of the sensing field, the density of sensors on the left side of the field is less than on the right side. Therefore, the sensing field is not fully covered by the sensors.

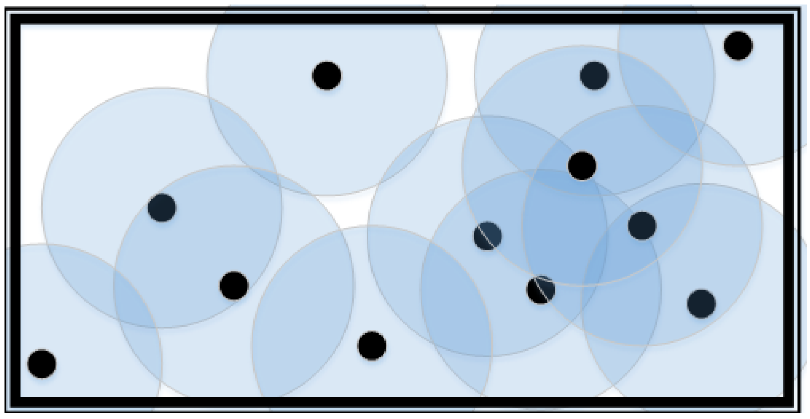


Figure 1.1 An example of area coverage by a randomly deployed sensor network.

In order to increase the coverage area, redundant sensors are deployed. The redundancy makes the sensor networks have higher density than normal ad-hoc networks. However, it is proved by [5] and [6], that just increasing the sensor density cannot provide coverage with 100% probability. Also, it is costly to maintain the large-scale, high density sensor networks. Therefore, some other approaches have to be taken in order to avoid these problems and still improve coverage after the sensors are randomly deployed.

Nowadays, with the development of Micro-Electro-Mechanical systems (MEMS), sensors have become so small that they have shrunk to the point where they can be carried by pet-sized, mobile robots. Also, certain wireless sensor networks utilize mobile robotic platforms for moving the sensors in a controlled manner. Therefore, instead of increasing the sensing density in a sensor network, the mobile sensors can be used for increasing the coverage.

In addition to the coverage problem of the randomly deployed sensor networks, energy consumption is also a major concern for mobile wireless sensor networks. In most applications, the lifetime of a system is critical to its effectiveness, especially for mobile sensor networks whose mobility systems consume more energy than the other components and processing devices. Energy consumption is a primary constraint for wireless sensor network nodes because most are self-powered by definition. Among the different parts of an individual network node, batteries are the only source of power for its entire life. Some applications are limited to the most essential components and do not have the option of recharging batteries. All of the node actions, such as, sensing, communicating, computing, and moving consume energy. Thus, once the battery or a sensor runs out of power, the sensor node is not useable anymore. This can degrade the quality of service for the entire sensor network.

In this dissertation, we will develop algorithms that relocate the sensors to improve the coverage of randomly deployed sensor networks, while also minimizing the sensor energy consumption so as to prolong the lifetime of the sensor networks.

There is also another concern that complicates the redeployment problem: the robustness of sensor networks. Once deployed, it is expensive and impractical, if not impossible, to replace unusable sensors in applications as the ones described above. Therefore, if an individual sensor node dies, it will affect the entire performance of the sensor network. The death of a sensor node can be attributed to various causes, such as a dead battery, physical damage in the field due to environmental forces, or being destroyed by enemy. If a sensor that covers a very sensitive area dies and no other sensor can cover that area, the sensor network fails its mission of distributing the sensors effectively.

1.2 Research Objectives and Contributions

The objective of the dissertation is to develop relocation algorithms for mobile wireless sensor networks that optimize both coverage and energy consumption. The specific tasks of our research are as follows:

1. **Develop distributed algorithms that optimize the coverage and energy used for carrying the sensors.** These algorithms can be performed by the sensor network without the use of a powerful central node. Therefore, the algorithms must not contain heavy calculations and also should not involve much communication overhead. The algorithms should also consider these three situations:

a) All sensors have very accurate localization systems on them.

b) None of the sensors have any localization systems on them.

c) All the sensors have localization systems, but with errors of the localization output.

Different technologies and methods have been developed and implemented for self-localization of wireless sensor nodes. However, in some environments such as indoors, there is still no accurate localization algorithm for wireless sensor nodes. All these cases need to be analyzed and a different set of algorithms need to be developed for them.

2. **Develop centralized algorithms for optimizing the coverage of sensor network and energy consumed for the sensors.** These algorithms can have higher computation requirements and be performed by a central node. The algorithms are only applicable when central nodes exist in the wireless sensor networks. Also, the use of a central node with high computational abilities should also provide better performance than the one provided by distributed algorithms.
3. **Develop algorithms using range-adjustable sensors to provider better coverage and also to save energy by shortening some of the sensors' sensing radius.** These algorithms are distributed algorithms that only deal with the sensors range but do not relocate them. They can be used in conjunction with the proposed and other existing relocation algorithms to provide energy savings.
4. **Enhance all the developed algorithms with the property of self-healing.** Whether a sensor network is capable of functioning well throughout its entire lifetime highly depends on its overall robustness. Algorithms with self-healing can increase the robustness of the system. The algorithms are designed such that when a sensor node dies, due to a drained battery or some physical condition change, the other nodes will adjust their status and react properly to minimize the influence of the sensor node's death on the sensor network's entire performance.

5. **Ensure our algorithms exhibit rapid convergence.** In most monitoring applications, reaction time is very important. The algorithms are designed to have a fast convergence rate when the sensors are first deployed and also when self-healing processes must take place.
6. **Evaluate our proposed algorithm.** The performance of our algorithms depends on area coverage, energy consumption, and the convergence speed. Different initial deployments are tested to ensure the convergence of algorithms. Also, we compare and evaluate the performance of our developed algorithm with the existing solutions.

In our work, seven different optimization algorithms are developed, including three distributed algorithms and three centralized algorithms for sensor relocation and one distributed algorithm for sensing range adjustment. All of the algorithms focus on both optimizing coverage with a limited number of sensors and prolonging the lifetime of the entire sensor network. However, this focus is considered differently in various applications and conditions, such as, if a central node exists or whether the sensors can self-locate. These algorithms should be capable of working with both indoor and outdoor monitoring applications, in harsh environments and disaster areas, and in real-time. Simulation results show that in all the different conditions described above, the corresponding algorithms optimize coverage of the sensor network within a reasonable cost of energy.

1.3 Methodology

In this dissertation, analytical, as well as computational, methods are used.

Firstly, we build the mathematical model for our optimization problem, which involves the maximization of the coverage and the minimization of the energy consumption. Different models related to the problem have been used in our analysis, and we simplify the mathematical model and optimize the solutions. Circle packing and Monte Carlo methods are used to analyze the coverage of the sensor networks.

Secondly, we compare our candidate solutions of the optimization problem with the ideal case of optimal solution and try to generate the solution close to the ideal. Different approaches have been developed in order to achieve the optimum solution.

Thirdly, we develop centralized solutions to the optimization problem. We developed multi-objective Genetic Algorithm and Particle Swarm optimizations algorithms to search for global optimization solutions. Both of these algorithms are heuristic optimization algorithms that will avoid local maxima or minima but provide globally optimal results. The optimum trade-offs are achieved between the conflicting objectives of better coverage and less energy usage.

Fourthly, simulations are done in the Matlab platform. Different random initial deployments are implemented in the simulations and the results are analyzed with probability theory. The simulations are repeated a significant number of times in order to analyze the results by statistical methods.

1.4 Dissertation Outline

The remainder of the dissertation is organized as follows. Chapter 2 discusses the related work. Chapter 3 describes the basic models and theories applied in our work. Two different sensing models are described with respect to coverage calculation. Models and theories related to energy savings include the energy models of the mobility components and the sensing power model of the sensing components, including range adjustable sensors.

Chapter 4 describes three distributed optimization algorithms: one for sensors with self-location devices, one for sensors without self-location devices, and the other for sensors with inaccurate self-location devices. Simulation results show that in all three cases, the coverage can be optimized. The first algorithm optimizes the average relative distances and provides fast convergence. The second algorithm works without localization system for sensors and provides optimized coverage by a trade-off of energy consumed for moving the sensors. The third algorithm works well when the localization systems of sensors are not accurate.

Chapter 5 discusses our centralized optimization algorithms. Three algorithms are described. The first is based on multi-objective, Genetic Algorithm. The second is based on traditional Particle Swarm Optimization. In the third algorithm, we hybridize the distributed algorithms into a Particle Swarm Optimization and generate a Hybrid PSO, which works uniquely for the sensor relocation problem. The first two centralized algorithms optimize the tradeoffs between coverage and energy consumption of the mobile sensor nodes. The hybrid PSO optimizes the lifetime of sensors and also optimizes the sensing coverage to be very close to 100%.

Chapter 6 discusses a novel sensing range adjustment algorithm that provides near to 100% coverage and also minimizes redundantly covered areas. This algorithm cannot be used for relocating the sensors but can be used cooperatively with our relocation algorithms and provide energy savings for the wireless sensor networks.

Chapter 7 summarizes the conclusions and provides recommendations for future research.

CHAPTER 2

RELATED WORK

This chapter discusses the related work and the literature view. We separate the related optimization algorithms into two groups with respect to their method of computation: distributed or centralized. After a summary of the existing algorithms, we discuss other related terms such as the self-location techniques in wireless sensor networks, the Voronoi diagram, the range-adjustable sensors, and the obstacle avoidance.

2.1 Distributed Optimization Algorithms for Sensor Redeployment

In the random deployment of sensor networks, distributed algorithms have been used. The most commonly used distributed optimization algorithms for coverage optimization of mobile wireless sensor networks are the virtual force related algorithms [7]-[11].

A potential field method was introduced by Howard, et al., [7], to optimize the coverage of sensor networks. Specifically, in [7], all sensors and obstacles create a virtual potential field. Therefore, every sensor is a point enacted upon by the combined potential field created by all the other sensors and obstacles. Each sensor will be impacted by a composition force from the virtual potential field. The interactive force of a particular potential field of another sensor or obstacle is directly proportional to the square of distance to that sensor or obstacle. The forces can be calculated as:

$$F_n = -k_n \sum_i \frac{1}{r_i^2} \cdot \frac{\vec{r_i}}{r_i} \tag{2.1}$$

$$F_o = -k_o \sum_i \frac{1}{r_i^2} \cdot \frac{\vec{r_i}}{r_i} \tag{2.2}$$

where F_n and F_o represents the force due to other nodes and obstacles correspondingly, $\vec{r_i}$ is the directional vector pointing for the center sensor to other sensors or obstacles and r_i is its magnitude. It can be seen from the above equation that both forces have a negative sign which means there are only repulsive forces. Sensors will be assigned weights so that they can move according to the virtual forces applied to them as they would in a real physical situation. Virtual frictional effects are also applied so that the algorithm can converge faster. It has been proved that equilibrium state can be achieved in a sensing field with boundaries. As a result, the sensors in the sensing field will rearrange themselves into a more uniform distribution after being randomly deployed, and also, the network's coverage can be significantly increased.

Sheng, et al., [8] also used the potential field algorithm. In their work, the potential field algorithm was used to relocate the sink nodes, which were carried by mobile robotic devices. After the relocation of the mobile sink nodes, more static sensor nodes could communicate to the sink nodes so that the quality of performance is increased.

Yi Zou, et al., [9], introduced a virtual force algorithm. The major difference between [7] and [9] is that [9] has both repulsive and attractive forces between sensors, whereas [7] only has the repulsive force. The force depends on the distance between sensors and their relative position. The forces can be represented as follow:

$$\overrightarrow{F_{ij}} = \begin{cases} (k_A(d_{ij} - d_{th}), a_{ij}) & if\ d_{ij} > d_{th} \\ 0 & if\ d_{ij} = d_{th} \\ (k_R / d_{ij}, a_{ij} + \pi) & if\ d_{ij} < d_{th} \end{cases} \tag{2.3}$$

where d_{ij} is the distance between sensors and a_{ij} is the direction from the center node to other sensor nodes, and d_{th} is a distance threshold that has a close relation to the sensors' sensing range. The objective of this algorithm is also to achieve better overall coverage by relocating the mobile sensors.

Different force models have been derived, including the impact of hotspots and obstacles model by Li et al.,[10]. A ranking system has also been applied to the force model, so that the algorithm can be used for varying application requirements.

Wong et al.,[11] improved the virtual force algorithms by applying a back-off delay time in the virtual force algorithm. The back-off delay time is used such that the sensors relocate themselves one-at-a-time in each round of movement. Thus, each sensor will have the most updated position information of the other sensors, including the movement of previous sensors within the current round. In this way, the sensors can move less than if they use the aged location information.

It should be pointed out that the virtual force related algorithms described above require that each sensor knows the exact or relative positions of all other sensors in the network. In other words, location information for sensors is required, and must somehow be communicated throughout the network.

Other deployment algorithms for randomly distributed mobile sensor networks were proposed by Wang et al.,[12], Pac et al.,[13], and Chang, et al.,[14]. These algorithms require that each sensor knows only the relative locations of sensors that are in its communication range, or its neighborhood.

Specifically, Wang et al.,[12], used Voronoi diagrams to detect areas that are not covered, or the sensing holes, and determined the way the sensors should move to cover those areas. Three different optimization algorithms have been developed: vector based optimization (VEC), Voronoi diagram based optimization (VOR), and Minimax optimization. The VOR and Minimax have similar performances that are better than the VEC when sensor nodes have low density. These algorithms also used a virtual movement method. With this method, each sensor calculates the future movements of all neighboring sensors and tracks these virtual movements to predict their future locations using the virtual locations. This prediction technique is repeated several times, and then the sensor moves only once. By using the virtual movement method, the total distance of sensor relocation is greatly decreased, along with the energy consumed.

Furthermore, Pac, et al., [13], used a fluid model for sensor relocation. In this algorithm, the author compared the mobile sensor network to a fluid model. The sensors are treated as fluid elements in a continuous medium, which represents the unknown sensing field. This algorithm can optimize the sensing coverage.

Chang, et al. [14] developed a density control coverage optimization algorithm. Each sensor divided its surrounding area into eight, hexagonal subareas, according to the relative direction. Then, the sensor density of each subarea was calculated, according to the number of sensors and the area of obstacles in that subarea. Then, the sensors moved toward the subarea that has the least density.

The deployment algorithms above are not designed to converge fast. However, the convergence speed is an important factor in real time applications.

2.2 Centralized Optimization Algorithms for Sensor Redeployment

The effectiveness of the deterministic deployment of a sensor network is a function of how efficiently it uses the sensor nodes to cover the desired area without any uncovered spots. This problem is related to the Art Gallery problem[15]. The Art Gallery problem is that of positioning the cameras of an art gallery so that the least amount of camera sensors can cover the entire art gallery. This problem is an optimization problem that is Non-deterministic, Polynomial-time hard. The video cameras in the Art Gallery problem are very similar to the sensors in wireless sensor networks. The main difference is that the video cameras have no limit of visual range, as they can see as long as no wall or obstacle blocks their line of sight. However, the sensor nodes always have a limited sensing range.

2.2.1 Genetic Algorithm

Genetic algorithms (GAs) are a type of evolutionary, optimization algorithm. Evolutionary optimization algorithms were first introduced by Barricelli in 1957[16]. GAs are heuristic search algorithms that come from the idea of natural evolution[17][18]. They take the concept of chromosomes, inheritance, mutation, and crossover in natural evolution. In a genetic algorithm, the input variables are treated as chromosome vectors. Different groups of initial variables will be generated. The genetic algorithm evaluates the values of the fitness function related to the input variables and keeps the chromosomes of the best fitness values. The GA simulates mutation and crossover of the chromosomes and always keeps the best values. So the best chromosomes are always inherited. GAs are very useful in obtaining the global maximum without being trapped in a local optimum. This is because the initial population covers the entire solution space, so that local optimized values will be avoided. They are suitable for solving nonlinear optimization problems and for finding the global optimization value of a fitness function.

Genetic algorithms can be used to solve optimization problems with NP-hard complexity. Also, deterministic deployment algorithms seek to prolong the lifetime of sensor networks by minimizing energy consumption. Some previously developed algorithms have used GAs to obtain optimum deployment of sensor networks.

Jourdan, et al., [19] used a Multi-Objective Genetic Algorithm to optimize the coverage and lifetime of an entire wireless sensor network. In the sensor network in [19], there is only one sink node. All sensors need to transmit data to the sink node directly or through multi-hop communication. Data transmission is the main concern of energy consumption in [19]. So with different communication ranges, the data transmission will have different number of hops, changing the energy consumed. The work presented in [19] deals with the conflict of the goals of minimizing energy consumption and maximizing coverage and uses a Multi-Objective Genetic Algorithm to find the optimal tradeoff between coverage and lifetime. Different optimum layouts are obtained with respect to different sensing-to-communication range ratios.

Jia, et al.,[20] deals with the scheduling problem for redundant sensors to prolong the lifetime of wireless sensor networks. The two objectives of their fitness function are to maximize the coverage and minimize the area covered by more than one sensor. They introduced a non-dominated sorting genetic algorithm (NSGA-II) to solve the optimization problem and minimize the number of sensors that need to be “on” to cover the maximum area. The other sensors are turned off so that they can save their energy and can be used when other sensors die.

Wang, et al.,[21] focused on improving the problem of selecting dynamic sensor nodes, which was first introduced by Burne, et al.,[22]. The problem is similar to the problem described in [20] but it involves the energy of each sensor in real time, dynamically. The objectives are to optimize the coverage ratio and prolong the lifetime of the entire sensor network. A Hopfield network (NH) is used to reduce the search space of the GA. Compared to the GA, this NH-GA converges faster.

Also, sensing range adjustment and coverage optimization were performed in[23]. The work presented in[23] used the Voronoi Diagram to determine the approximate sensing range of each sensor and then used the GA to optimize the utilization of power.

Liang, et al.,[24] performed the optimization work for wireless sensor networks in 3D environments. In this work, a force driven genetic algorithm was introduced to decide the location of the sensors in order to obtain an optimized coverage and also consider energy savings.

GAs are suitable for optimizing the deployment positions of sensor nodes, scheduling of groups of sensor nodes, and optimizing energy usage such as optimizing the sensing range and communication. GAs are very powerful optimization tools for the wireless sensor networks. However, the main restriction of GAs is they have very high computation complexity; thus, they require powerful CPUs. A normal sensor node usually does not have the ability to do so, and a central node with powerful computational capabilities is always required for GAs.

2.2.2 Particle Swarm Optimization

Particle Swarm Optimizations (PSOs) was first introduced by Kennedy, et al.,[25] in 1995. The idea of a PSO comes from the natural behavior of birds seeking food. When a group of birds are looking for food together, each bird will first look around in the area near itself. Each bird will communicate to the other birds where it finds the most amount of food near its area. Thus, all the birds can know which areas have the most amount of food in their entire, collective feeding area. Birds will continue looking for food in nearby places, especially where the most amount of food had been found in the entire area. PSO algorithms simplify this concept of organized labor. Similar to GAs, a group of candidates will be generated in PSOs from the entire space. Each candidate is a set of vectors that contain the variables related to the problem. The group of candidates will evolve to the combination of personal best fitness and the group global best fitness. This is like the process of the birds seeking food.

Compared to GAs, the PSO has the advantage that it is easier to program and implement, and it has less parameters to control. The PSO has been widely used in wireless sensor networks[26], such as node and base station positioning[27], node localization[28][29][30], data aggregation[31][32], and energy aware clustering[33][34].

PSO has been used in coverage and energy optimizations for non-mobile wireless sensor networks. Wang, et al.,[35] used a parallel particle swarm optimization algorithm for optimizing the energy used by sensors to track targets. The parallel PSO is used for maximizing the coverage and minimizing the energy consumption by turning off the sensors that are far away from the targets. The simulation results showed siginificant improvement in energy efficiency.

The PSO has also been used for relocating mobile sensor nodes[36]-[39]. Bai, et al.,[36], used PSO to optimally relocate sensors that are initially randomly deployed. The objective of their PSO is to optimize the coverage of the sensor network with least amount of sensor movements. Specifically, they studied two different cases of sensors mobility. In the first case, the sensors can move unlimited distance. In second case, the sensors modeled with limited mobility, which means they can only move within a certain maximum distance. The PSO algorithm improved the network's coverage in both cases, especially in the limited mobility model. Also, the energy spent for the sensors' movement was greatly reduced.

Wang, et al.,[37], developed a new PSO algorithm, called VFCPSO, which combines the virtual force (VF) algorithms and co-evolutionary particle swarm optimizations (CPSO). In this algorithm, virtual forces are used to update the evolving velocity of each candidate solutions and different, cooperating swarms. The algorithm optimizes the coverage of a network that contains both static and mobile sensors. Compared to the traditional PSO, the VFCPSO can perform better with increasing dimensionality of the optimization problem and decreasing the computation time of the VFCPSO. Simulation results showed that VFCPSO provides a 10% coverage increased compared to the traditional PSO. Huang and Lu [38] also applied the PSO to the coverage optimization of hybrid, wireless sensor networks.

Li, et al., [39] imported selection and mutation to the PSO and named their algorithm Particle Swarm Genetic Optimization (PSGO). Their algorithm aims at improving the density of nodes. Mobile nodes move to cover the uncovered areas (coverage holes). Their work demonstrated that by relocating only 5 of 100 sensors, the algorithm could increase the coverage by 6%. However, these algorithms do not focus on energy savings and prolonging the lifetime of sensor networks with limited energy resources.

2.3 Self-localization in Wireless Sensor Networks

Self-localization is the ability to know one's location with respect to some agreed upon reference, and it is very important for wireless sensor networks, since most applications require knowledge of where the data is coming from. It is also important to the coverage optimization algorithms, most relocation algorithms discussed above require knowing at least the relative locations of sensors, and they cannot function if the sensors have no self-localization system. Therefore, sensor localization has become an active research topic in recent years.

There are two types of sensor self-localization methods; one is Fine Grained Localization and the other one is Coarse Grained Localization. The first type usually has less error in the localization results than the second, but it usually involves either higher computational complexity or hardware cost.

A typical Coarse Grained Localization algorithm for wireless sensor networks is the Distance Vector hop (DV-Hop) method introduced by Niculescu[40]. In this algorithm, the distances between nodes are estimated by the least number of hops they need to communicate, where a hop is a communication link used between sensors that form part of a larger segmented transmission path. Since neither the hop distance between each sensor pair is constant, nor are their associated directions always the same, this algorithm can only estimate the approximate area that other nodes are in, and not their exact locations. Different algorithms are developed to optimize the original DV-Hop. Li, et al.,[41] developed a weighted DV-Hop self-localization scheme that can increase the accuracy of the traditional DV-Hop. Other improved algorithms are [47-51].

Bulusu, et al., [47] introduced a reference-point based, low-cost self-localization algorithm for wireless sensor networks. In this algorithm, base stations, or other types of infrastructures, are located at grid points. They send out beacon signals including their location information to nearby sensors. These sensors use this information to estimate their own location by assuming they are at the centroid, or average location, of all the stations from which they receive beacon signals. If a sensor node can hear from reference nodes located in (x_1,y_1), (x_2,y_2),… (x_n,y_n), the sensor node will self-locate itself as:

$$\begin{cases} x_{est} = \dfrac{1}{n}\sum_{i=1}^{n} x_i \\ y_{est} = \dfrac{1}{n}\sum_{i=1}^{n} y_i \end{cases} \tag{2.4}$$

where (x_{est},y_{est}) is the estimation location decided by the sensor. Thus, this method does not require the sensors to know or use the strength of the beacon signal, as the exact distance is not calculated. The accuracy depends on the density of the reference points and the distances between them. Furthermore, the accuracy of this method increases as the density of the reference points increases

Xu, et al.,[48] introduced a coarse localization algorithm based on received signal strength indicators (RSSI). In this algorithm, three base stations are placed in different corners of a sensing field that, again, send out beacon signals to nearby sensors. A signal strength map is generated and stored by the individual sensor nodes. In the map, there are also discrete reference points at locations of pre-determined signal strength, which are known to the sensor nodes. The individual sensors will compare the received signal strengths of the three base stations with that of the reference nodes and assign to themselves the location of the reference points with the most similar values.

In the Coarse Grained Localization schemes described above, there is not much calculation involved. However, the Fine Grained localization is different. Fine-grained localization algorithms can be classified into three types: received signal strength (RSS) based, time of arrival (TOA) based, and direction of arrival (DOA) based.

In RSS based localization algorithms, radio propagation model equations are used, as the received signal strength between sensors or base stations are directly related to the distance between them. The main process of an RSS based localization algorithm is the determination by the sensors of their locations. First, each sensor receives the beacon from the reference nodes (base stations etc). Then, it calculates the distance to each reference node. Finally, it solves the distance equations to get its own location. The main obstacle of this type of algorithm is that the parameters, especially that of the path loss exponent (also called distance-power gradient), in the equations are not easily obtained due to variations in the environment of sensing field and the uncertainty of channel noise in lognormal fading. Thus, different algorithms are developed to optimize the solutions of the equations and minimize the localization error.

Li, [49] developed an RSS based localization algorithm for unknown distance-power gradients. He used the gradient method to minimize the estimation error. Simulation results show a significant advantage of this algorithm over ones with fixed distance-power gradient values when the distance-power gradient is not accurate. Similar works are also done in [50]

MacDonald, et al., [51] analyzed the RSS based localization algorithm with an isotropic transmitter. They also estimated the distance-power gradient to be 2, which is the free space value. They also use the gradient method to optimize their results. They tested their algorithm with localized, cell phone base stations along Lake Shore Drive in Chicago.

Other relevant RSS-based works are included in the following. Shi, et al., [52] use the steepest descent method to refine the node position of RSS and minimize the influence of the noise. Yao, et al., [53] used Particle Swarm Optimization to increase the localization accuracy. Jia, et al., [54] used Genetic Algorithm to perform the optimization. Amutha, et al., [55] developed a hybrid algorithm that first uses the hop count to approximate the positions of the sensor nodes and then uses RSS to refine the result.

TOA is another distance based localization algorithm. In this type of algorithm, the distance is calculated using the time differences between when the beacon signal is transmitted and when it is received. Radio waves are propagate very fast, thereby requiring a very accurate clock to determine the locations precisely. Therefore, the TOA algorithms have a high cost. The global positioning system (GPS) is a typical application of the TOA localization[56]. Mobile devices use the signal received from satellites to calculate the distance between the satellites and themselves. The satellites also send their own location information, so that the mobile devices can calculate and estimate their own locations. A detailed survey about TOA based localization algorithms is presented in [57]. Other TOA algorithms can be found in [58-60].

DOA and AOA (angle of arrival) algorithms are used for localization sensor nodes. In these algorithms, individual sensors have the ability to tell the direction of beacon signal they received. This usually requires antenna arrays or some other complicated hardware using directional antennas. Sensors use the location information of the reference nodes and the angle, or direction, at which they received the beacon signal to solve triangulation equations and determine their locations [59-62].

When sensor networks are used indoors, accurate self-localization becomes significantly more difficult. This is primarily due to multipath fading, or signal degradation when radiated through dense materials, such as metal and concrete found in buildings. Various improvements have been implemented to reduce the noise in indoor environments. Pu, et al., [65] used time-series filters to eliminate the noise in RSS localization algorithms. Wu, et al., [66] used radio map filters to deal with noise. Chen, et al., [67] analyzed the obstacles' influence on the localization systems and derived an error-based improved RSS localization algorithm.

The localization error of wireless sensor networks used indoors is at mimimum 2 meters, in existing manufactured products. This is tolerable if there are not any walls or obstacles that must be taken into account. However, if a sensor node is near a wall, which is about 20 cm thick on average, the previously discussed localization schemes are not able to determine on which side of the obstacle the sensor is located. With this level of uncertainty, most relocation schemes described before cannot function properly for indoor sensor networks.

2.4 Voronoi Diagram

A Voronoi diagram is a method of decomposing an area. Assume there is a set of *N* nodes deployed in an area without any obstacles, the Voronoi diagram will divide the entire area into *N* subareas, and each subarea has a single node inside it. The characteristic of Voronoi diagram is that each subarea is composed of the area closest to the node inside it, as opposed to the other nodes. The generation of the Voronoi diagram requires the location information of all nodes. Voronoi diagrams are very useful in the coverage problem of wireless sensor networks. If each sensor can cover its own Voronoi subarea, the entire sensing field can be covered.

Voronoi diagrams have been used for detecting uncovered areas. Aziz, et al., [68] used PSO to optimize the coverage of wireless sensor networks. Voronoi diagrams are used for calculating whether there are any uncovered areas, so as to calculate the fitness function. Meguerdichian, et al., [69] examined the best-worst case coverage in target tracking in wireless sensor networks. A Voronoi diagram is used for the worst case coverage analysis. The worst case coverage happened when the target traveled along the edge of Voronoi subarea. Boukerche, et al., [70] optimized the centralized Voronoi diagram approach and reduced the computation for coverage problem in wireless sensor networks.

Wang, et al., [12], also used a Voronoi diagram for uncovered area detection in wireless sensor networks. After detecting the uncovered area, sensors move toward that direction, and thus increasing the coverage. Wang, et al., [12], also developed a scheme in which the sensors are always moving to the center of the Voronoi subarea, so that the sensors will have a larger chance to cover the entire Voronoi subarea.

Voronoi diagrams have also been used to optimize the coverage of wireless sensor networks. Li, et al., [71] used Voronoi diagram to assist in reducing the number of sensors needed to cover the maximum area for directional sensors. Lee, et al., [23] applied the Voronoi diagram on wireless sensor networks to save the energy when range adjustable sensors are used. The Voronoi diagram can help choose the sensing range that is shortest without decreasing the coverage of the entire sensor network.

2.5 Range-Adjustable Sensors

Range-adjustable sensors can change their sensing range by controlling their power. This type of sensor can be used to save energy by shortening the sensing range. There are now some sensors available with range adjustable capabilities, such as some motion detection sensors and photoelectron sensors. Motion detector sensors are very typical sensors that can be found in safety monitoring or in harsh environments.

Wu, et al., [72] studied a coverage problem with range-adjustable sensors. They attempted to minimize the number of sensors and maintain the coverage, in order to lower the energy consumption. Different energy models have been proposed to describe the relationship between the sensing range and consumed energy. A scheduling model has been proposed to achieve the objectives.

Cardei, et al., [73] introduced an energy saving method using range-adjustable sensors. They grouped the sensors into Adjustable Range Set Covers (AR-SC), such that each set can cover the entire desired area. A scheduling scheme was also proposed, so that the energy can be saved and the lifetime of the entire wireless sensor network can be optimized.

Dhawan, et al., [74] also developed algorithms for a similar problem as the one handled by [73]. A mathematical model was proposed and the approximation algorithm of Garg-Konemann was used to determine the approximate solutions. Simulations showed that their algorithm can increase the lifetime of sensor network by four times compared to the one of [73].

2.6 Energy Consumption

As we described in the first chapter, a common mobile sensor has several components, such as the CPU, memory, battery, sensing device, and mobile device. These components will need to use energy from a battery. It will be very helpful to know the energy consumed by each of the components.

Pei, et al., [75] did a survey on hardware platforms for wireless sensor networks. They summarized the sensor nodes hardware for different applications. They analyzed the energy consumption for different communication protocols and microprocessor platforms. Stojcev, et al., [76] summarized the energy consumed in different types of sensors such as a GPS, a barometric sensor, and a humidity sensor. They also compared the power of some common radio modules. Mei, et al., [77] presented a mobile device for carrying wireless sensor nodes. They analyzed the energy consumed when a sensor node moves straight or turns.

2.7 Summary

This chapter summarized both distributed and centralized algorithms for coverage optimization of wireless sensor networks. We also described some key terms related to coverage optimization: the localization of sensors is critical for the coverage optimization algorithm; the Voronoi diagram is helpful for coverage calculation and energy savings; range-adjustable sensors can also be used for energy saving. Lastly, we discuss the energy consumption of different sensors' components.

CHAPTER 3

MODELS AND THEORIES

This chapter describes the models and theories that are related to the coverage optimization algorithms. The coverage of wireless sensor networks is related to the sensing model used. Different sensing models are introduced in this chapter, and the corresponding methods for coverage calculation are described in the following. We also explain the ideal model for sensor deployment for optimizing coverage, without any uncovered areas in between the sensors. In order to optimize the energy usage in individual wireless sensor nodes of the network, we also describe the energy models for mobile sensors and range adjustable sensors.

3.1 Sensing Models

There are two types of sensors in the real world. The first type of sensor only is concerned about the data at the point where it is located, such as temperature, humidity, and pressure sensors. The second type of sensor has a certain range in which it can detect, such as motion detectors and video camera sensors. In our discussion, we mainly focus on the second type of sensors, because they are more commonly used for area monitoring and safety surveillance.

In the real world, the sensing range of sensors may be irregular due to the obstacles in the environment, such as rain and snow. Fei, et al., [78] researched the irregular sensing range due to the existence of obstacles in the real world and proposed an α-shape range detection model.

However, in order to simplify the analysis and calculation, the sensing range of each sensor is always assumed to be a circular area. Commonly, there are two different types of sensing models used for simulating the performance of sensors [9]: the binary and probability models.

The difference between binary sensing model and probability sensing model is that, in the probability model, if a target is in a certain area, the target may be detected with a certain probability between 0 and 1. However, there is no such area in binary model. In binary model, a target can either be detected or not. This difference between the models is illustrated in Figure 3.1. The notation used is explained below.

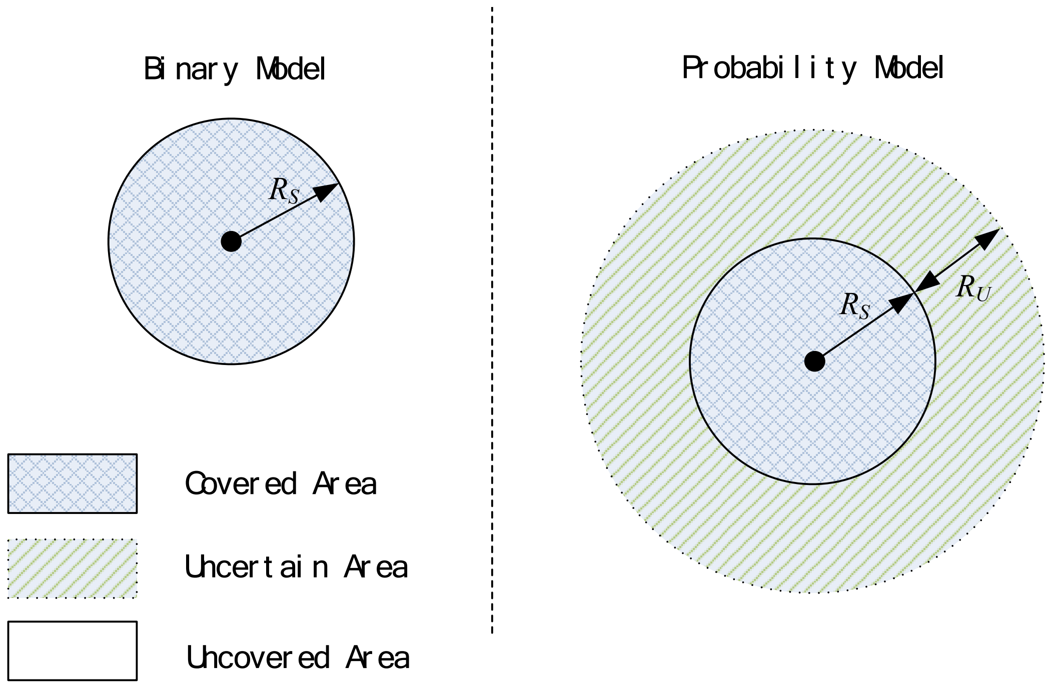


Figure 3.1 Difference between binary sensing model and probability sensing model.

3.1.1 Binary sensing model

If there is a sensor node *S* at location (x_s, y_s), we assume that the sensing range of the sensor *S* is a circular area with radius R_S and centered at (x_s, y_s). R_S is called the sensing radius of node *S*.

In the binary model, the sensor *S* is able to detect the target inside its sensing range with a probability 1, and it is not able to detect any target that is outside of its sensing range. Thus, in the binary model, a sensor can detect the target with a probability of 1 if the distance between the target area and the sensor is less than the sensing radius R_S. However, if the distance between the target and sensor is farther than R_S, the sensor will have zero probability of detecting it.

Assume a target *T* is located at coordinate (x_t, y_t), the probability that the target *T* will be detected by sensor *S* in binary model can be expressed in the following equation:

$$p_{sb} = \begin{cases} 1 & D_{TS} \le R_S \\ 0 & Otherwise \end{cases} \tag{3.1}$$

where p_{sb} represents the sensing probability in binary model, D_{TS} is the distance between target *T* and sensor *S* which can be calculated as:

$$D_{TS} = \sqrt{(x_s - x_t)^2 + (y_s - y_t)^2} \tag{3.2}$$

There is no transitional period in the binary model. A slight difference in locations may result in a totally different detection output. The binary model is the most simplified model for sensing.

3.1.2 Probability sensing model

In the probability model, unlike the binary model, there is a transitional period between when a sensor absolutely can and cannot detect a target. There will be a certain area for each sensor, in which the sensor cannot tell if a target can be detected. In that area, the target will have a probability to be detected between 0 and 1.

In the probability model, there are two critical distances for a sensor. The first one is R_S, which is the same as the one in the binary model. If the distance between the target and the sensor is less than R_S, the target can be detected by the sensor with a probability of 1. The second critical distance is R_U, which stands for the uncertain range. If the distance between the target and the sensor is in the range between R_S and R_S+R_U, the probability that the target will be detected by the sensor is related to the distance between them. If the distance between the target and sensor is farther than R_S+R_U, the target will not be detected by the sensor. The mathematical expression of probability model is as followed:

$$p_{sp} = \begin{cases} 1 & D_{TS} \le R_S \\ e^{-\lambda a^{\beta}} & R_S < D_{TS} \le R_S + R_U \\ 0 & R_S + R_U < D_{TS} \end{cases} \tag{3.3}$$

where $a = D_{TS}$ - R_S, and λ and β are constants related to the sensors' hardware properties. The relationship between the probability model, λ, and β is shown in Figure 3.2. In the figure, R_S is set to be 10 meters and R_U is set to be 25 meters.

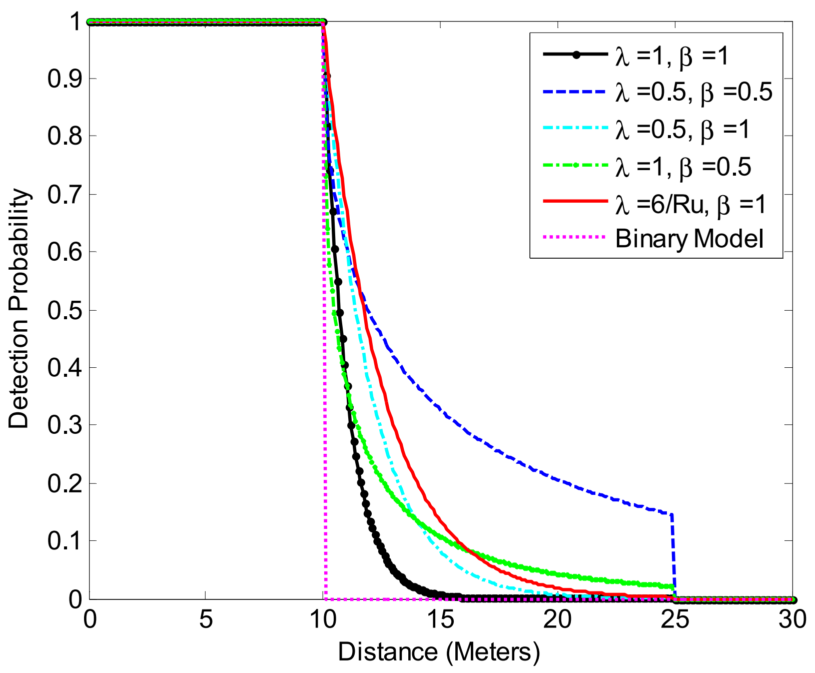


Figure 3.2 Detection probability in Probability model and Binary model.

It can be seen in Figure 3.2, that when the distance between the target and the sensor is closer than R_S, both binary model and probability model give a detection probability of 1. When the distance is longer than R_S and shorter than R_S+R_U, the binary model gives a 0 probability of being detected, but the probability model will have a gradually decreasing probability. When the distance is longer than R_S+R_U, both of the sensing models will give a zero detection probability.

Different parameter values result in different transitions regions of the decreasing probability. As it can be seen, when λ and β are both equal to 1, the probability drop from 1 to 0 in about 6 meters, and when λ and β are both equal to 0.5, the probability has not dropped to 0 after 15 meters. In our algorithm, we want to choose the value of λ and β that can make the probability drop from 1 to 0 gradually in the entire range of R_U. Assume β = 1, then λ can be calculated by this equation:

$$e^{-\lambda R_U} = \varepsilon \tag{3.4}$$

where ε is a small positive value.

We calculate that when λ is equal to $6/R_U$, ε will be equal to 0.0025. It can be seen in Figure 3.1, when $\lambda = 6/R_U$ and $\beta = 1$, the detection probability will drop from 1 to 0 continuously in the entire range of R_U. We will use these values in our simulation in the following chapters.

The probability model is more realistic than the binary model and the binary model is the simplified version of probability model when R_U is zero. However, the binary model is much easier to analyze.

3.2 Coverage Calculation Model

In Section 3.1, the two sensing models have been described. As the binary model is a special case of the probability model, we can just use one coverage calculation model that fits both of the sensing models.

The definition of coverage ratio ($R_{Coverage}$) is the ratio of area that can be covered by sensors cooperatively ($A_{Covered}$) over the entire sensing field (A_{Total}):

$$R_{Coverage} = \frac{A_{Covered}}{A_{Total}} \tag{3.5}$$

The maximum coverage ratio is 1. In the rest of this dissertation, we will use a percentage to describe the coverage ratio.

An area is covered if it can be covered by at least one sensor node or by the joint detection of several sensors. Assume there are N sensor nodes in the entire sensing field, the joint detection probability for a certain area can be calculated as below:

$$P_{Cover} = 1 - \prod_{s=1}^{s=N}(1 - p_s) \tag{3.6}$$

where p_s can be calculated by equation (3.1) and (3.3) corresponding to the model adopted.

In the binary model, P_{Cover} will be either 1 or 0, which means the area will be covered or not. However, in the probability model, P_{Cover} will be any value from 0 to 1. A threshold (P_{th}) is required for judging if an area is covered. If P_{Cover} is bigger than P_{th}, the area is considered to be covered, otherwise, the area is not covered. The value of P_{th} depends on the application requirement.

Due to the random positions of sensors, the area they cover will be irregular. One method for calculating an irregular area is the Monte Carlo method. The Monte Carlo method is a statistical method for finding numerical results for the problems that are hard to find or do not have analytical results. In the Monte Carlo method, random samples will be evaluated repeatedly. If a sufficient number of samples are evaluated, the results will be very close to the expected value of the outcome [79] [80]..

In the Monte Carlo method, the area of an irregular field can be calculated in the following procedure. First, a regular area that surrounds the target field is chosen. Second, we randomly pick a sufficient number of points in the regular area and judge if each is in the target field. Last, the coverage ratio is calculated with the number of points inside the target field over the number of all sample points. The area will be the coverage ratio multiplied with the area of the regular area.

In our algorithm, we did not sample randomly when calculating the coverage. Instead, we treated the sensing fields as a grid, and used each grid point as a sample point for calculating the coverage. The coverage ratio can be calculated by:

$$R_{Coverage} = \frac{n}{N} \tag{3.7}$$

where n is the number of grid points that meet the requirement that $P_{Cover} \geq P_{th}$. It is implied that if P_{th} is 1, the probability model is equivalent to the binary model.

The accuracy of this method depends on the distance between grids. In our simulations, the sensing field has been divided into 100 by 100 grid points, which means there are 101x101 = 10201 points in total. Therefore, the error in the coverage calculation for our simulations is less than 1%.

Examples of the coverage calculation in both models are shown in Figure 3.3. In figure 3.3, three sensor nodes A, B and C are deployed in a 20 meter by 20 meter square field. There are 21*21 = 441 grid points in total. The parameters we used here are: the sensing radius is 3 meters, the uncertain radius is 4 meters, and the coverage probability threshold is 0.5. The circles with radius R_S represent the real covered areas, with the binary model is applied. It can be seen that all the grid points have probability 1 to be detected. We count the number of nodes that have a detection probability 1, which is 79. So the coverage of binary model is 79/441 = 17.9%. The different color solid points are those that are covered when employing the probability model. The color bar on the right side indicates the probability. All other hollow points are places the probability model cannot cover. The number of grid points covered when applying the probability model is 95, so that the coverage ratio when we use probability model is 21.5%.

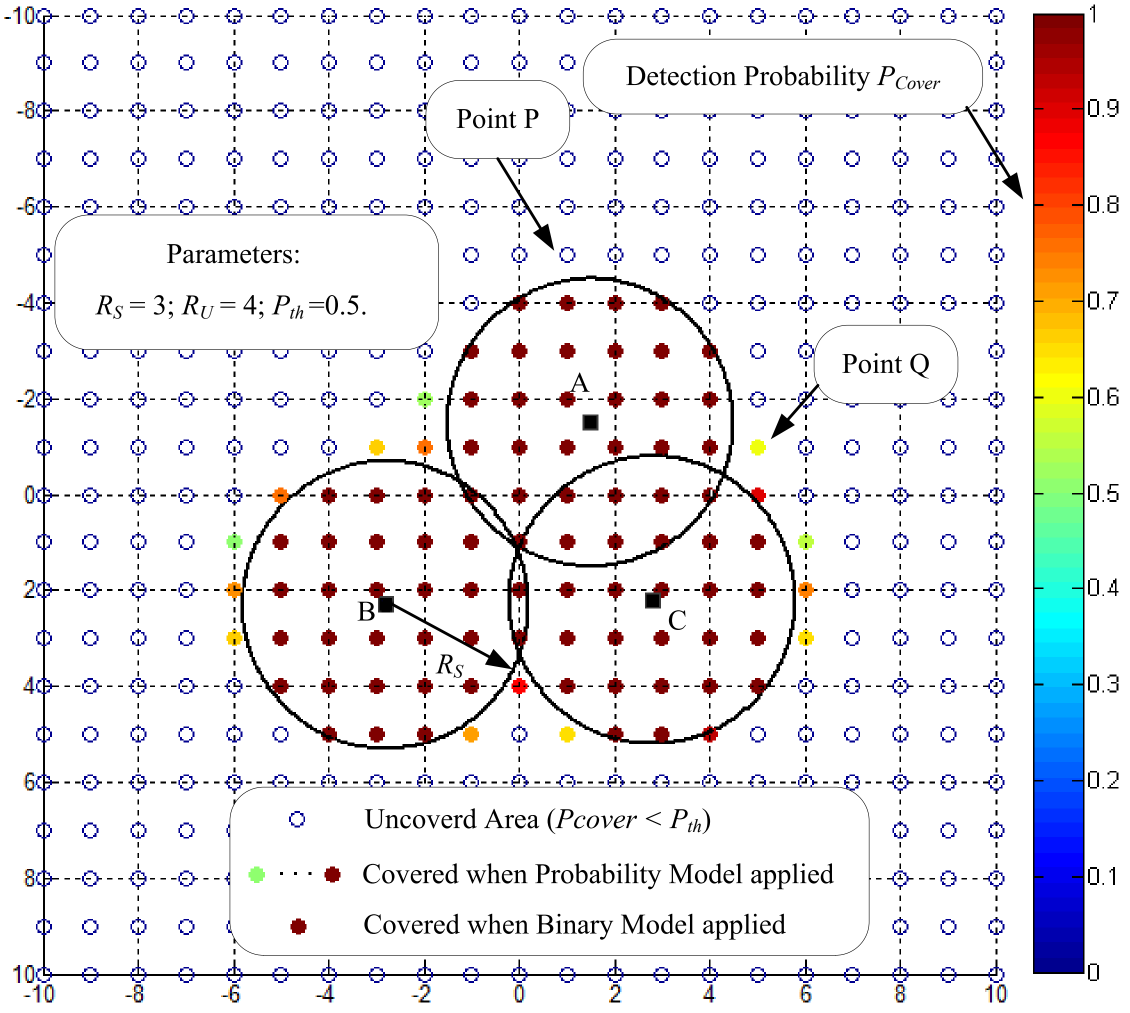


Figure 3.3 Coverage calculation for Binary model and Probability model.

Compare point *P* and *Q* in Figure 3.3 in probability model. They have similar distances to sensor node *A*; however, point *P* is not covered but point *Q* is covered. This is because *Q* is in the uncertain range of both sensor *A* and *C*. Therefore, the joint detection probability of *Q* becomes larger than 50%.

3.3 Ideal Coverage Model

Consider a sensing field without any obstacles or attractive areas inside. The deployment of sensors with maximum coverage can be achieved as in [81] and [82]. Boll, et al., [81] works on circular packing problems. However, in this work, they only deal with circle packing of adjacent sensors to each other so that a gap will be between each pair circles. Circles are not allowed to overlap. This is not suitable for analyzing the problem of sensor deployment, because in sensor deployment the sensors' covered areas can overlap.

Wang, et al., [82] analyzed the problem of seamless coverage in sensor networks. Optimal deployment of sensors is achieved when all sensors have the same equal sensing radius. The optimal layout is shown in Figure 3.4. This is an ideal coverage model for the binary sensing model. This can be used for the deterministic deployment of sensors when the boundary of sensing field has a regular shape and there are no obstacles in the sensing field.

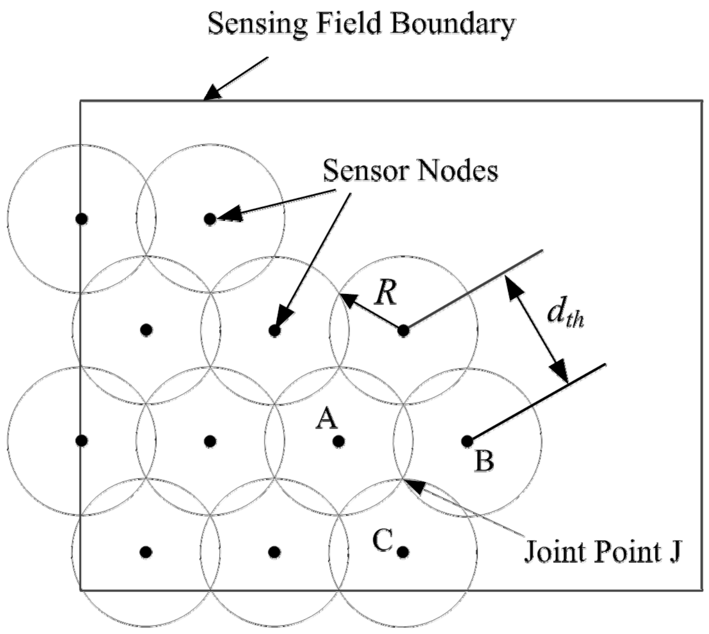


Figure 3.4 Ideal coverage model.

Figure 3.4 shows that in the ideal coverage model sensors are deployed regularly with same distance between sensors. The distance between sensors (d_{th}) can be calculated by the sensing radius:

$$d_{th} = \sqrt{3} \cdot R \tag{3.8}$$

a) Binary model

When the binary model is used, R will be equal to the sensing radius R_S, therefore, d_{th} can be calculated using R_S as below:

$$d_{th} = \sqrt{3} \cdot R_S \tag{3.9}$$

b) Probability model

When the Probability model is used, the joint point J will have a detection probability of P_{th} by the sensor nodes A, B and C. Using the same notations as previous, the radius R can be calculated by solving equation (3.3) and (3.6):

$$P_{Cover} = 1 - \prod_{s=1}^{s=3}(1 - p_s) = P_{th} \tag{3.10}$$

$$p_s = e^{-\lambda(R-R_s)^{\beta}} \tag{3.11}$$

Since the distances from sensor node A, B and C to joint point P have same distance, the single detection probability from A, B and C are the same. Equation (3.10) can be simplified to:

$$1 - (1 - p_s)^3 = P_{th} \tag{3.12}$$

$$p_s = 1 - \sqrt[3]{1 - P_{th}} \tag{3.13}$$

Also, we are using the parameters in Section 3.2 in which λ = -6/R_U and β = 1. R can be solved in the following way:

$$e^{-\frac{6}{R_U}(R-R_s)} = 1-\sqrt[3]{1-P_{th}} \tag{3.14}$$

$$R = R_S - \frac{R_U}{6}\ln\left(1-\sqrt[3]{1-P_{th}}\right) \tag{3.15}$$

Therefore, d_{th} can be calculated by:

$$d_{th} = \sqrt{3}\left[R_S - \frac{R_U}{6}\ln\left(1-\sqrt[3]{1-P_{th}}\right)\right] \tag{3.16}$$

Comparing equation (3.9) and (3.16) for binary model and probability model, when P_{th} equals to 1 or R_U equals to 0, equation (3.16) is equivalent to equation (3.9).

However, this optimal model is ideal because it can only be used for optimizing the deployment without obstacles. If obstacles are present in the sensing field, the problem becomes more complicated. No formulated algorithm is derived when different shapes and numbers of obstacles appear in the sensing field.

3.4 Energy Model for Mobile Sensors

Our algorithms aim at relocating sensor nodes to increase the overall network coverage ratio. In the relocation process, energy will be consumed by each sensor node as they move or turn.

When the weight of sensors and speed are constant, we consider the power consumed in relocating to be constant as well. Therefore, the energy consumed in moving the sensors (E_{dis}) is linear with respect to distance it traveled when acceleration is negligible. Also, every time a sensor node turns, energy will be consumed. The energy consumed for turning (E_{turn}) is linearly related to the angle sensor node turned. These can be summarized by the following equations:

$$E_{dis} = k_{dis} \cdot D_{total} \tag{3.17}$$

$$E_{turn} = k_{turn} \cdot A_{total} \tag{3.18}$$

where k_{dis} and k_{turn} are coefficients representing the energy consumption rate. D_{total} and A_{total} are the total distance travelled and the total angle turned by the sensor respectively. The total energy for moving the sensors will be the summation of the two types of energy listed above:

$$E_{moving} = E_{dis} + E_{turn} \tag{3.19}$$

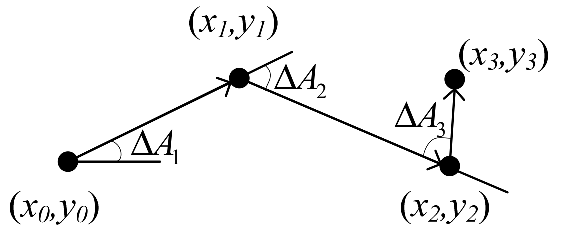


Figure 3.5 Moving energy calculation illustration

As shown in Figure 3.5, we assume a sensor is originally located in (x_0,y_0), and moves to location (x_n,y_n) by moving through (x_i,y_i) ($i= 1,\ldots n-1$). We also assume that all sensors start off facing the direction of positive x-axis. Also, a sensor can turn either clockwise or counter clockwise, and it will always choose the smaller angle to turn to its destination direction. D_{total} and A_{total} can be calculated by equation (3.20) and (3.21):

$$D_{total} = \sum_{i=1}^{n} D_i = \sum_{i=1}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2} \tag{3.20}$$

$$A_{total} = \sum_{i=1}^{n} \Delta A_i \tag{3.21}$$

Energy efficient motors have been used for designing the low energy mobile sensors [83][84]. Omni-directional small robots have been built by Reshko, et al., [85] which can be used in mobile sensors. In [85], three wheels are used in each robot, and the wheels are uniformly fixed along the edge of a round-shaped platform and pointing to three different angles 120^o apart. The locations of the wheels form a regular rectangle. In this way, the robot can move straight or turn omni-directionally. Mei, et al.,[86] provided the energy model for this type of omni-directional mobile robot. According to [86], the energy for a robot platform to move 1 meter is 9.34 Joules, when traveling at a constant speed of 0.08m/s. The energy for this three-wheel robot to turn 90 degrees is 2.35 Joules. We assume the sensors in the algorithm move and turn at a constant speed. Therefore, k_{dis} and k_{turn} can be calculated with the parameters provided by [86]:

$$k_{dis} = 9.34\left(Joules / Meter\right) \tag{3.22}$$

$$k_{turn} = \frac{2.35}{90} = 0.0261\left(Joules / Degree\right) \tag{3.23}$$

3.5 Energy Model for Adjusting Range

Range adjustable sensors are used because when they shorten their sensing radius, the power consumed will be lower thereby yielding energy savings.

Different energy models are used for analyzing the relationship between a sensors' sensing radius and its energy consumption, and they depend on the characteristics of the sensing device. Typical models are the linear, quadratic, cubical and quadruplicate models. The most common are the linear and quadratic models[87],[73].

For range adjustable sensors, there is always a maximum sensing radius that the sensors can achieve. We denote this maximum sensing range as R_{SMax}. When R_S is shorter than R_{SMax}, the power consumed has certain relationship with the sensing radius R_S.

In the linear model, the energy consumption of the sensor device is linearly related to the sensing radius:

$$P_l = k_l \cdot R_S \quad (R_S \leq R_{SMax}) \tag{3.24}$$

where P_l is the power consumed in sensing, and k_l is a device-dependent constant related to the power consumption rate.

Similarly, in the quadratic model, the energy consumption is depends quadratically on the sensing radius:

$$P_q = k_q \cdot R_S^2 \quad (R_S \leq R_{SMax}) \tag{3.25}$$

where P_q is the power consumed in sensing, and k_q is a constant for power consumption rate.

In the linear model, it is implied that the energy consumption is linear to the sensing radius. However, in quadratic model, the energy consumption is linear to the sensor's covered area.

For example, a sensing device consumes 4 Watts when it works with its maximum sensing radius of 20 meters. Then k_l and k_q can be calculated with equation (3.24) and (3.25):

$$4\,Watts = k_l \cdot 20\,Meters \tag{3.26}$$

$$4\,Watts = k_q \cdot (20\,Meters)^2 \tag{3.27}$$

So that k_l is 0.2 *Watts/Meter*, and k_q is 0.01 *Watts/Meter*^{2}. In this case, if the sensing radius is shortened to 10 meters, the power in linear model reduces to 2 Watts and the power in quadratic model reduces to 1 Watt. This example shows that when the sensing radius has reduces by half, the energy savings are 50% and 75% for the linear and quadratic energy models, respectively.

In our analysis energy savings, both the linear and quadratic energy models will be used.

3.6 Obstacle Model

Our algorithms will also consider the existence of obstacles. Obstacles can be walls, trees or any other physical body which is not part of the regular sensing field. Usually, obstacles will have an impact on wireless sensor networks in three different ways: communication, sensing, and mobility. In this subsection, the aspects of obstacles important to wireless sensor networks will be characterized.

1. Obstacles have no impact on the communication between sensors. The obstacles will not lower the signal strength, so that the sensors can communicate with each other even when they are on opposing sides of the obstacle.
2. Obstacles will block the sensing function. Sensors do not have the ability to cover the areas that are blocked by obstacles, even if the distance from that area to the sensor is less than R_S. The detection probability for an area blocked by an obstacle is 0.
3. A sensor can detect the existence of an obstacle. It can also tell the location and the shape of the obstacle if and only if the obstacle falls into the covered area of the sensor.
4. Obstacles will block the movement of the sensors. The sensors cannot move across any obstacles. With point 3 listed above, sensors can detect the existence of the obstacles, so they will be programmed not to hit the obstacles. If the destination of a

sensor is on the other side of an obstacle, it must go around the obstacle to reach its desired destination.

In our algorithms, we do not focus on how to avoid obstacles. We only listed the characteristics that are related to our relocation process.

3.7 Assumptions

Fine-tuned optimizations in the real world are designed by adjusting to various, small-scale non-idealities, such as irregular shape of sensing field, obstacles, and disruptions of communication between sensors. In order to simplify these problems, our algorithms assume the following:

1. All sensors are mobile and can move according to the direction and distance to which they are instructed. Also, each sensor has a compass so it can tell its direction and turn to the destination directions.
2. All sensors have a circular communication area. The radius of communication range is R_C. This means, if the distance between a pair of sensor is shorter than R_C, they can directly communicate with each other, otherwise, these two sensors cannot communicate directly. In order to provide the connectivity to the wireless sensor networks, R_C needs to be larger than R_S. The circularity of the communication area is not a requirement of the optimization algorithm implementation, but instead is merely for simplifying the problem.
3. The sensing field for wireless sensor networks needs to cover a clear boundary. The sensing field will not be of infinite area. For simplicity, a rectangular area is assigned to be the sensing field in our analysis and simulations. This is also applicable when

the ideal coverage model is derived. Also, the boundary is pre-known and can be programmed into the sensors before being deployed. In this way, the sensors will not exceed the boundaries of the sensing field.

There are some additional assumptions that deal with the type of sensing models used, the existence of obstacles, and the inclusion of self-localization devices, and centralized powerful nodes. These assumptions will be separately discussed with different algorithms.

3.8 Summary

In this chapter we described and defined the models related to our optimization algorithm, and we also presented the assumptions of our models. Our optimization algorithms focus on relocating sensors, in order to increase the coverage of WSNs and prolong their lifetime. We also described the coverage and energy related models. In the optimization of coverage, we described the binary and probability sensing models, the coverage model, and the ideal optimum coverage model for a field without obstacles. In the realm of energy savings, we described the energy model for mobile devices and range-adjustable sensors. The obstacle model is also described as certain situations do present obstacles in the sensing field.

CHAPTER 4

DISTRIBUTED OPTIMIZATION ALGORITHM DESIGN

In this chapter, three distributed optimization algorithms are introduced. Compared to the centralized algorithms, these algorithms require less computation, so that they can be applied in each sensor node. As there is no central node in control, sensor nodes work cooperatively to increase the coverage ratio of the entire sensor network in all algorithms. In the first two algorithms in this chapter, localization systems are required. However, in the third algorithm no localization systems are required. In these algorithms, sensors move more than one time and gradually increase the coverage. Furthermore, simulations and comparisons with other existing methods are performed.

4.1 Optimization with Average Relative Position

In this section, a distributed coverage optimization self-relocation algorithm using the average relative position between pairs of sensors is introduced. Both relative distance and direction will be used while performing the optimization. Therefore localization systems are required for applying this algorithm. Adjustable range sensors are used in order to minimize the cost of energy for sensing.

4.1.1 Assumptions

In addition to the general assumptions in Section 3.7, the following assumptions are made by this algorithm.

1. All sensors are identical in computational ability. The algorithm will be performed by each sensor and no central node will be used for computing the optimizations.

2. All sensors are equipped with localization systems. They also have pre-installed knowledge of the sensing field, such as the position of boundaries, and obstacles.

3. In this algorithm, only the binary sensing model is used.

4.1.2 Optimal coverage distance threshold estimation

As we describe in Section 3.3, there is an ideal coverage model for deployment of wireless sensors. Our algorithm uses the ideal coverage model to calculate a threshold for redplo볼yment of sensors.

Even though the coverage threshold can be calculate by equation (3.8), due to the adjustable range sensors, this equation is not applicable here. In this algorithm, we use the average area that should be covered by each sensor to estimate the sensing radius.

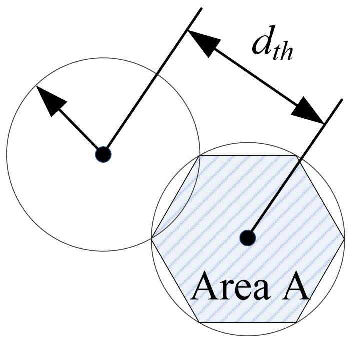


Figure 4.1 Distance threshold estimation.

Consider a field with area A_{total} that must be covered by N sensors. In this case, each sensor must cover an area of:

$$A = \frac{A_{Total}}{N} \tag{4.1}$$

The distance threshold can be estimated by:

$$d = \sqrt{\frac{2}{\sqrt{3}} A} = \sqrt{\frac{2}{\sqrt{3}} \frac{A_{Total}}{N}} \tag{4.2}$$

Sensors that are close to boundaries or obstacles will cover an area less than A, as described by equation (4.1). Therefore, in order to compensate for this area increase, A must be increased thereby increasing the distance threshold. In order to account for this, we assume a 10% increase in our threshold calculation, which leads to:

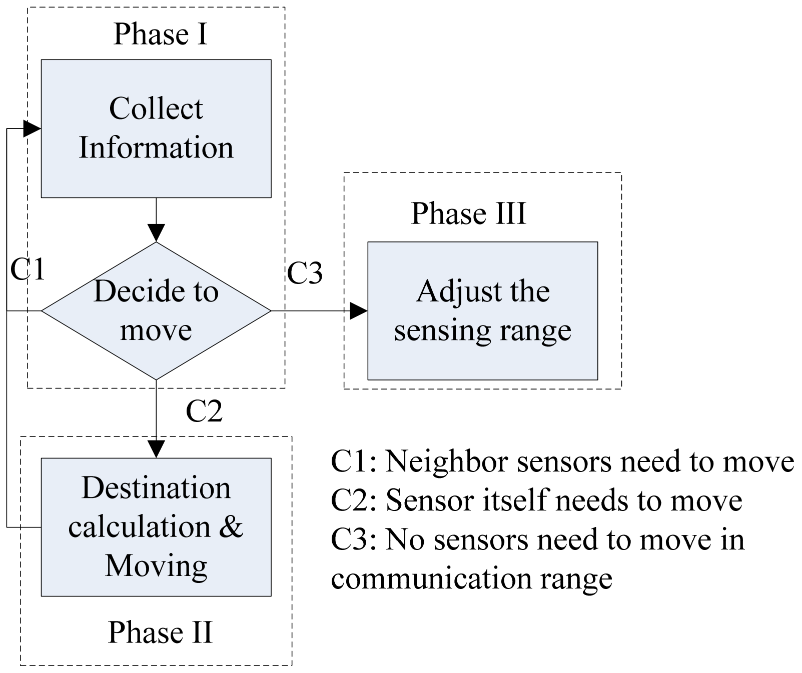
$$d_{th} = 1.1d = 1.1 \times \sqrt{\frac{2}{\sqrt{3}} \frac{A_{Total}}{N}} \tag{4.3}$$

In practice, the distance threshold depends on the shape of the sensing field and the position of obstacles. Therefore, equation (4.3) provides only an estimation for the distance threshold. Also, the distance threshold should not be greater than $\sqrt{3}\ R_{SMax}$. Our algorithm uses equation (4.3) to estimate the distance threshold when sensors are initially deployed in a field.

4.1.3 Algorithm description

The objective of our self-deployment algorithm is to relocate the randomly deployed sensors and perform a sensing range adjustment so that an optimized coverage is achieved with minimum consumption of energy. There are three phases in this algorithm. The first phase performs the decision making. The second phase moves the sensors to a new location. The third phase performs sensing range adjustment.

In the first phase, each sensor locates itself and broadcasts its position information to the other sensors within its communication range. Sensors that can communicate with each other are called neighbors. Therefore, each sensor has its own communication neighborhood. This neighborhood depends on the communication range, R_c, and the layout of the field (i.e. obstacles). Sensors obtain the position information of other sensors through the broadcast process, and then they decide to move or not according to the location information they collected. In phase two, sensors move only if the movement criteria are met. After the sensors have moved, they broadcast their locations again and the algorithm restarts from phase I. The flow diagram of the algorithm is shown in Figure 4.2.



Phase I
Collect Information
Decide to move
C1
C3
Phase III
Adjust the sensing range
C2
Destination calculation & Moving
Phase II
C1: Neighbor sensors need to move
C2: Sensor itself needs to move
C3: No sensors need to move in communication range

Figure 4.2 Flow diagram of algorithm.

a) Phase I: Information collection and relocation decision

Initially, each sensor locates themselves by GPS or other method. Each sensor constantly broadcasts its location information and receives the location information of other sensors. Every sensor generates a local map with this information. If a sensor finds any obstacles or sensing field boundary inside its sensing range, it will also add this information into its local map. Each sensor decides if it should move or not based on the movement criteria below and it broadcasts its decision to the neighboring nodes. Specifically, the movement criteria are as follows:

(a) A sensor must move away from other sensors if there is at least one sensor at a distance closer than 0.9 d_{th};

(b) A sensor must move closer to other sensors if *criterion (a)* is not met and no more than one sensor is located at a distance closer than 1.1 d_{th};

(c) A sensor does not have to move if neither of the above criteria are met.

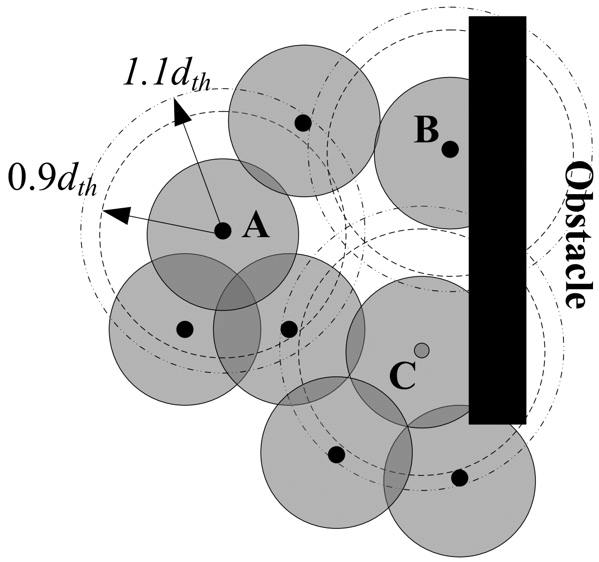


Figure 4.3 Example for moving criteria.

An example of the moving criteria is shown in Figure 4.3. Eight sensors and one obstacle are in the sensing field. The gray area is covered by the sensors. In this example, sensor *A* meets *criterion (a)*, since there are two sensors that are at distances from node *A* that are less than 0.9d_{th}. Sensor *B* does not meet *criterion (a)* since no sensors are at a distance less than 0.9d_{th} from it. Also, no sensor is at a distance that is less than 1.1 d_{th} from sensor *B*. Thus, sensor *B* meets *criterion (b)*. Finally, sensor *C* does not meet neither *criterion (a)* or *(b)*; therefore, it meets *criterion (c)* and does not need to move.

b) Phase II: Destination calculation and movement

Following the movement criteria mentioned above, a sensor may need to move closer to or further away from other sensors. In order to decide its destination, a sensor must estimate the direction and the travel distance of the movement.

1) Calculation of Movement Direction

The direction of movement is calculated for each sensor using the direction of the gradient of the average distance between the sensor and its neighbors:

$$grad(\bar{d}) = \frac{\partial \bar{d}}{\partial x} i + \frac{\partial \bar{d}}{\partial y} j \tag{4.4}$$

where $\bar{d}$ is the average distance of a sensor from other surrounding sensors. The gradient direction corresponds to the direction that a variable changes the fastest; therefore, we can use it to optimize the relocation of the sensors.

In *criterion (a)*, the direction of movement, θ_a, is equal to the gradient's:

$$\theta_a = \angle grad(\bar{d}_a) = \angle\left(\frac{\partial \bar{d}_a}{\partial x} i + \frac{\partial \bar{d}_a}{\partial y} j \right) \tag{4.5}$$

where d_a is calculated by averaging the distance of only the sensors that are within a distance d_{th} from the sensor:

$$\bar{d}_a = \frac{1}{n_1} \sum_{i=1}^{n_1} d_i = \frac{1}{n_1} \sum_{i=1}^{n_1} \sqrt{(x-x_i)^2 + (y-y_i)^2} \quad d_i < d_{th} \tag{4.6}$$

and n_1 is the number of sensors that met the requirement $d_i<d_{th}$.

In *criterion (b)*, sensors need to move closer to other sensors (i.e., the sensors must move toward the direction that makes $\bar{d}$ decrease). Since the angle of the gradient indicates the direction toward which $\bar{d}$ increases the fastest, the direction of movement, θ_b , is opposite from the direction of the gradient:

$$\theta_b = \angle grad(-\bar{d}_b) = \angle\left[-\left(\frac{\partial \bar{d}_b}{\partial x} i + \frac{\partial \bar{d}_b}{\partial y} j\right)\right] \tag{4.7}$$

The average distance d_b represents the average distance of a sensor from the sensors inside its communication range:

$$\bar{d}_b = \frac{1}{n_2}\sum_{i=1}^{n_2} d_i = \frac{1}{n_2}\sum_{i=1}^{n_2}\sqrt{(x-x_i)^2 + (y-y_i)^2} \quad d_i < R_c \tag{4.8}$$

where n_2 is the number of sensors that met the requirement $d_i<R_c$.

2) Calculation of moving distance

The aim of the algorithm is to evenly distribute sensors so that the distance between them is approximately d_{th}. The travel distance for a sensor is set equal to the difference between the distance threshold d_{th} and the average distance calculated by equation (4.6) or (4.8) as:

$$d_{move} = | d_{th} - \bar{d} | \tag{4.9}$$

where $\bar{d} = \overline{d_a}$ for *criterion (a)* and $\bar{d} = \overline{d_b}$ for *criterion (b)*. If $d_{move}<0.1d_{th}$, we set $d_{move}=0$ according to the 10% margin set in the criteria above.

In order to prevent sensors from moving back and forth, we also introduce a control mechanism. Specifically, sensors record the direction they moved toward last time and compared it to the new direction, if the difference between the two times is greater than 90%, the sensor will not move until the next round.

3) Obstacle avoidance

It is assumed the sensors can detect the location of obstacles in their sensing range. Therefore, in our algorithm, the sensors avoid obstacles by maintaining a certain distance from them. Also, sensors will move along the side of obstacles instead of choosing a random direction in the next round.

c) Phase III: Sensing range adjustment

Each sensor can generate a local Voronoi diagram based on the information it receives from its neighboring sensors. Each sensor adjusts its sensing range according to its distance to the vertices of its Voronoi subarea. The sensing range adjustment can be done using different strategies that in turn will provide different coverage. Voronoi diagrams also consider the existence of boundaries and obstacles. (see Figure 4.4)

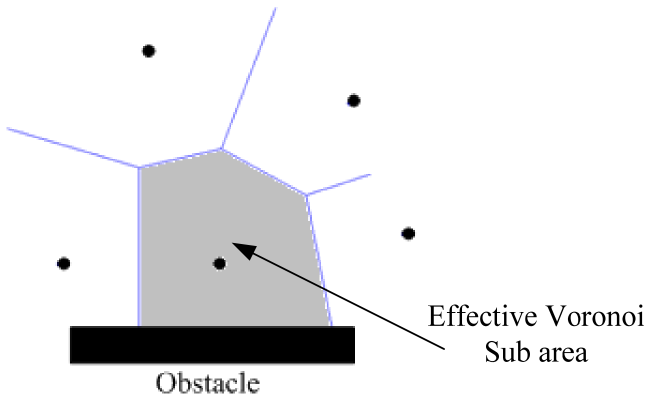


Figure 4.4 Voronoi diagram.

The different strategies for sensing range adjustment are described in what follows. The distance from the center node of each Voronoi subarea to the subarea's vertices is denoted as dv_i, where $i=0,1,\ldots,n$, and n is the number of vertices:

Strategy 1- Maximum radius:

In this strategy the sensor's sensing radius is set to be equal to the distance from the sensor to the farthest subarea's vertex:

$$R_s = Min\left[R_{SMax}, Max(dv_i)\right] \tag{4.10}$$

In this way, each sensor will cover the entire effective Voronoi subarea it belongs to. Therefore, the entire sensing field will be covered. This strategy will provide the maximum coverage among our three strategies.

Strategy 2- Second Largest radius:

Sometimes the angle between two edges of subareas is very sharp. This always happens on the vertex that has the furthest distance from the sensor node (see Angle V in Figure 4.5. In this situation, if sensors still choose the maximum radius as their sensing range, they will consume more energy to cover a small uncovered area. In order to provide a better trade-off between maximum coverage and energy consumption, the algorithm in this *Strategy 2* chooses the second largest radius.

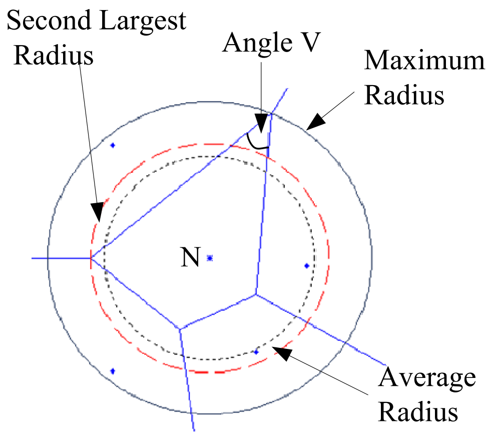


Figure 4.5 Sensing range adjustment strategy illustration.

In *strategy 2*, the sensing range of each sensor is set equal to the second largest distance from the sensor to the vertices of its subarea:

$$R_s = Min\left[R_{SMax}, SecondMax(dv_i)\right] \tag{4.11}$$

<u>*Strategy 3- Average radius:*</u>

The sensing range of each sensor is set equal to the average distance of the sensor to the vertices of its subarea.

$$R_s = Min\left[R_{SMax}, \frac{1}{n}\sum_{i=1}^{n} dv_i\right] \tag{4.12}$$

From all the above strategies, *strategy 1* will provide the largest coverage using the largest sensing range for the sensors. According to the energy model for adjusting range, in Section 3.5, the sensing range cannot exceed the maximum sensing radius R_{SMax}.

4.1.4 Simulation and results analysis

There are three parts in our simulation analysis. The first part involves the evaluation of our algorithm's convergence and energy consumption with respect to travelled distance. The second part involves the comparison of the performance of the three strategies for sensing range adjustment. The third part involves the evaluation of the self-healing capability of our algorithm.

The sensing field for all our simulations is assumed to be a 100 meters by 100 meters. We set the grid resolution to be 1 meter, so that a 101 $\times$ 101 grid is formed and used by our analysis. The sensors used here are assumed to have a communication range of R_c = 55 m, and a maximum sensing range of R_{SMax} = 25 m. These values are typically in practical sensor networks [75]. For example, a commonly used communication protocol for wireless sensor networks is Zigbee, which uses a communication range of 75 m. However, considering that the sensing field here is only 100 m by 100 m, we set the communication range to be 55 m so that not all of the sensors can communicate with all others. Also, a sensing range of 25 m is a typical sensing range for a motion detector sensor. Finally, we assume that there is no limit for the distance that a sensor node is allowed to move.

The distance threshold can be calculated based on equation (4.3) as:

$$d_{th} = 1.1 \times \sqrt{\frac{2}{\sqrt{3}} \frac{A_{Total}}{N}} = 1.1 \times \sqrt{\frac{2}{\sqrt{3}} \frac{100 \times 100}{20}} = 26.4m \tag{4.13}$$

a) Convergence and energy consumption evaluation

The performance of a relocation algorithm is evaluated here based on the following:

1) The convergence rate, which corresponds to the number of rounds required to reach convergence;
2) The coverage provided;
3) The energy consumption that is closely related to the travelled distance by all sensors.

Three different initial deployments are used to test our relocation algorithm. There are 20 sensors in each case. In the first case, all sensors are randomly deployed in the center part of the sensing field. In the second case, all sensors are randomly deployed in the entire area of the sensing field. In the third case, twenty sensors are equally divided into two groups that are randomly deployed near the left and right boundary.

The simulation results for the convergence of the sensing coverage are shown in Figure 4.6. In all three cases, the potential filed algorithm, [7], along with the parameters described in [8], is applied, in order to compare its performance with the performance of our algorithm. It can be seen that both algorithms converge to 98% to 99% coverage. However our fast converging relocation algorithm requires approximately 5 rounds versus the potential field algorithm that requires approximately 15 rounds. This means that our algorithm can save 70% of the energy used by calculations required for the relocation of sensors.

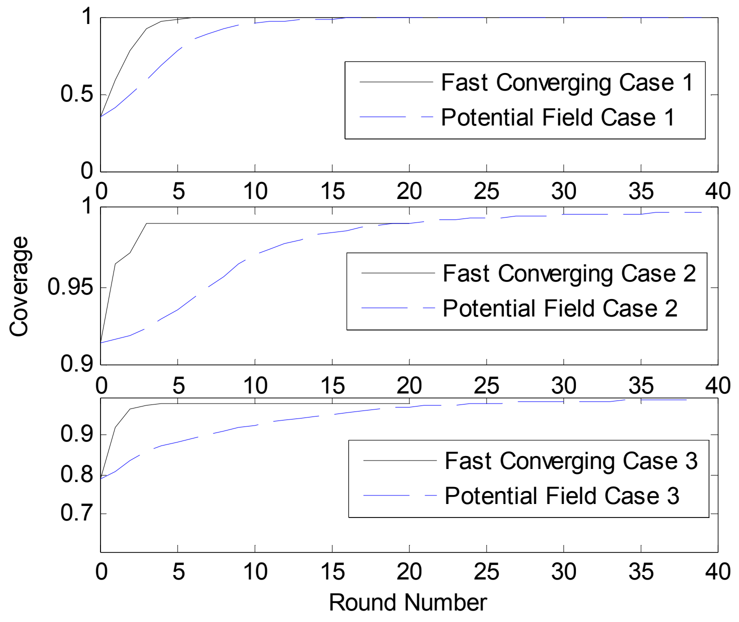


Figure 4.6 Sensing coverage convergence.

Also, the energy consumption for each algorithm is calculated according to the average travelled distance. We use the energy consumption model that was presented in [86]. The robots have a speed of 0.08 m/s, while the acceleration is 0.2 m/s. The energy consumption for constant speed is 9.34 J/m. The energy consumption can be obtained using these values.

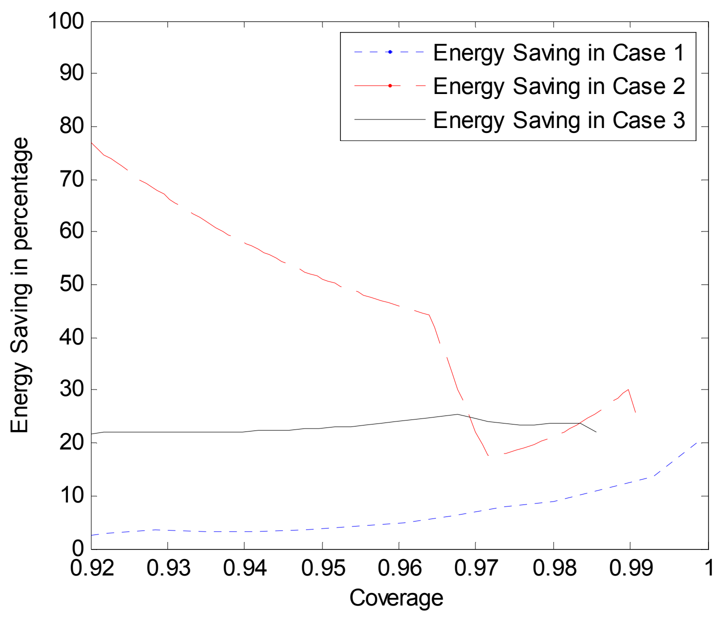


Figure 4.7 Savings in energy consumption.

Figure 4.7 shows the percentage of energy savings of our algorithm compared to the potential field algorithm. For, our three cases, the energy savings are plotted, when the algorithms have reached at least a coverage larger than 92%. The percentage energy savings are calculated by:

$$\text{Percentange Energy Savings} = (E_F - E_P) / E_P \tag{4.14}$$

where E_F is the energy consumed by fast our converging algorithm, and E_P is the energy consumed by the potential field method. Figure 4.7 shows that in all three cases, our fast converging algorithm provides significant energy savings. Specifically, our algorithm converges at its maximum coverage using 20% to 25% less energy than the one used by the potential field algorithm.

Also, the performance of our algorithm is tested in two cases with obstacles. Both cases use the same initial deployment by initially placing the sensors near the center of the sensing field (see Figure 4.8). However, in the first case, there is one obstacle

(obstacle 1; see Figure 4.8), and in the second case, there are two obstacles (obstacles 1 and 2; see Figure 4.8).

The initial coverage of the cases with one and two obstacles is 30% and 29%, respectively. The coverage convergence of our algorithm is shown in Figure 4.9 for both cases with obstacles and for the case without any obstacles. When an obstacle appears, the algorithm converges in approximately 10 to 12 rounds, versus 5 rounds for the case with no obstacles. Also, the maximum coverage achieved by our relocation algorithm drops from 100% in the case of no obstacles to around 94% in both cases with obstacles. The theoretical maximum coverage in the cases of obstacles is 99% and 98% for the cases with one and two obstacles, respectively.

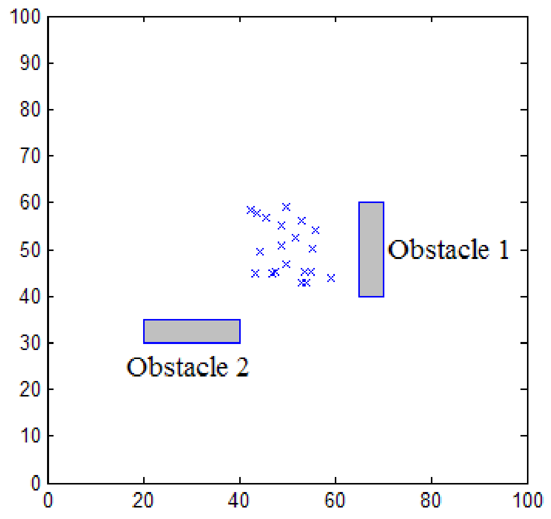


Figure 4.8 Initial deployment of sensors in a field with obstacles.

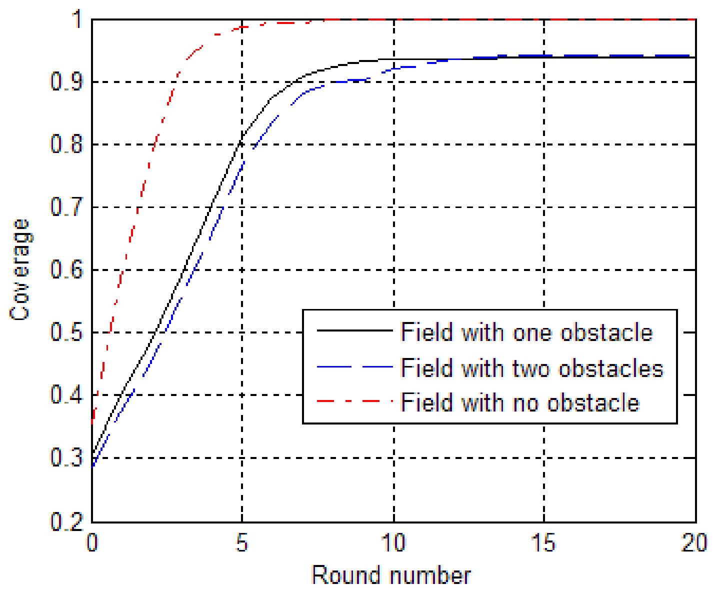


Figure 4.9 Sensing coverage convergence.

The results of a typical simulation for the case with two obstacles are shown in Figure 4.10. Figure 4.10 shows the movement traces of the sensors and the final coverage achieved by our algorithm. It can be seen that the obstacles block the spreading of sensors in the field. The sensors move along the edges of the obstacles and find ways to spread out. However, in the center there is still more density of sensors than the other areas. Potential field algorithm is not implemented here since [8] did not provide detailed how to handle obstacles.

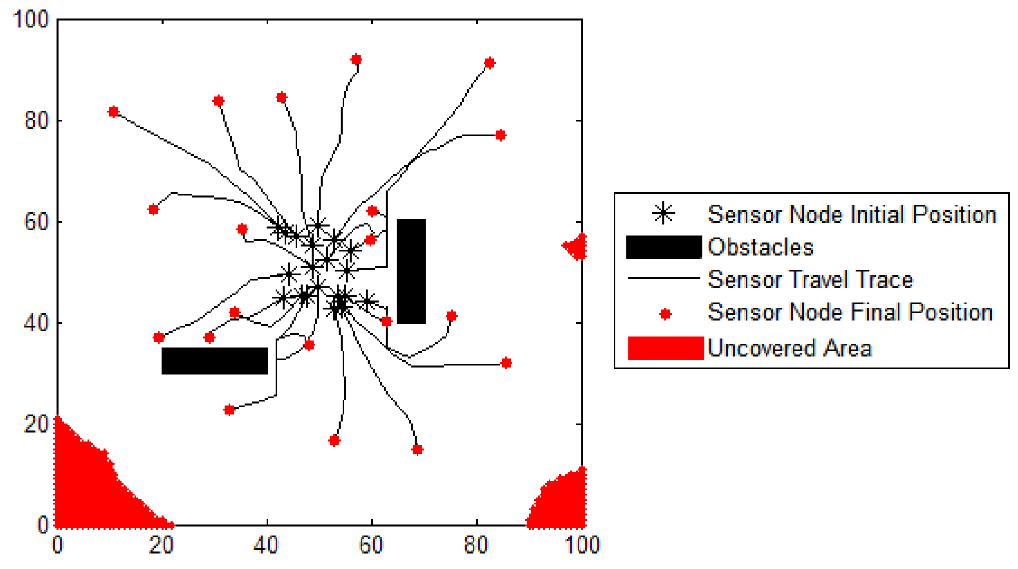


Figure 4.10 Simulation for a sensing field with two obstacles.

b) Sensing range adjustment strategies

After the relocation algorithm has converged, range adjustment is applied according to the three strategies discussed above. Specifically, this is done for the three cases examined in the previous subsection: no obstacle, one obstacle and two obstacles.

Two different energy consumption models have been introduced in Section 3.5: (a) linear energy consumption model, and (b) quadratic consumption model. In the linear energy consumption model, the energy consumption of sensors depends linearly on the sensing range. In the quadratically consumption model, the energy consumption of sensors depends quadratic on sensing range. In our simulations, we calculate the energy savings using both models.

The results are shown in Table 4.1. The energy savings of each range adjustment strategy are calculated by comparing the energy used by these strategies to the energy required when the sensors use their maximum sensing range of 25 meters. For example, if a sensor adjusts its range to 20 meters, its energy savings is (25-20)/25=20% in linear model and $(25^2-20^2)/25^2 = 36\%$ in quadratic model. In Table 4.1, E_{LSAVE} is used to denote the percentage of energy savings in linear model and E_{QSAVE} for quadratic model.

Table 4.1

SENSING RANGE ADJUSTMENT RESULT

		NO OBSTACLE	ONE OBSTACLES	TWO OBSTACLES
WITHOUT ADJUSTMENT	AVE R_S	25 M	25 M	25 M
	COVERAGE	99.9%	94.1%	94.0%
ADJUSTMENT STRATEGY 1	AVE R_S	19.2 M	19.3 M	19.8 M
	COVERAGE	99.9%	94.1%	94.0%
	E_{LSAVE}	23.2%	22.8%	20.8%
	E_{QSAVE}	41.0%	40.4%	37.3%
ADJUSTMENT STRATEGY 2	AVE R_S	17.2 M	17.5 M	18.1 M
	COVERAGE	99.1%	93.2%	93.3%
	E_{LSAVE}	31.2%	30.0%	27.6%
	E_{QSAVE}	52.7%	51.0%	47.6%
ADJUSTMENT STRATEGY 3	AVE R_S	15.8 M	16.3 M	16.4 M
	COVERAGE	96.0%	90.2%	88.7%
	E_{LSAVE}	36.8%	34.8%	34.4%
	E_{QSAVE}	60.6%	57.5%	57.0%

The results are shown in Table 4.1. Specifically, Table 4.1 illustrates that all three strategies yield energy savings by shortening the sensing ranges. Strategy 1 can achieve the same coverage with the case, where no range adjustment was performed (i.e. every sensor uses its maximum sensing range), and provide 41.0% energy savings. Strategy 2 provides energy savings of 50% but its coverage reduces by 1% from the coverage achieved by strategy 1. Strategy 3 provides energy savings of 57%, but its coverage decreases by 5% compared to the coverage achieved by strategy 1.

c) Self-healing process

In this section, we examine the self-healing performance of our algorithm. Therefore, we randomly remove five sensors from a field with two obstacles leaving 15 sensors in the field. After five sensors have died, the coverage of the whole network drops from 94% to 89%. Then, our relocation algorithm is applied, and it converges to a coverage of 96.5% after 8 rounds. The movement traces of the sensors are shown in Figure 4.11. It can be seen that the active sensors redistribute themselves and cover the coverage hole created by the dead sensors.

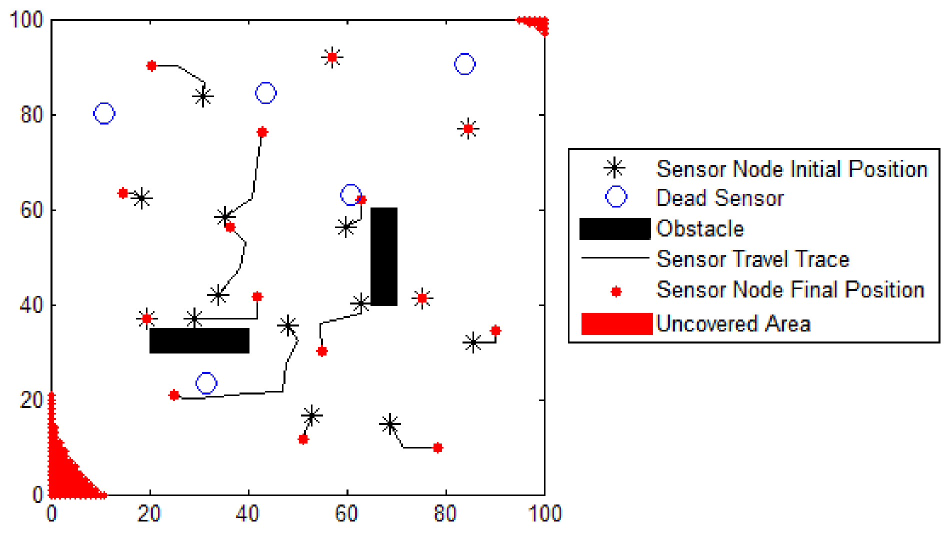


Figure 4.11 Movement traces and final coverage of sensors after self-healing process.

4.1.5 Conclusions

This section describes a fast converging relocation algorithm for wireless sensor networks, which relies on the average relative distance between the sensors. The simulation results show that this algorithm converges significantly faster than the potential field deployment method for fields with or without obstacles. The algorithm also provides energy savings by requiring less travelled distance than the conventional potential field method. Furthermore, the algorithm performs well in field with obstacles. With the sensing range adjustment it also provides significant energy savings. Different strategies for sensing range adjustment lead to different energy savings and coverage. A strategy should be used based on the requirements of each application. Finally, the algorithm has self-healing capabilities thereby improving the coverage of sensor networks after some sensors have stopped working.

4.2 Optimization with Weighted Relative Distance

This section introduces a distributed self-relocation algorithm using weighted relative distance. It is appropriate for mobile wireless sensor networks with random initial deployments. The algorithm aims at changing the relative distance of the sensors in different directions. Unlike the previous algorithm, it only uses part of the relative distances to calculate for optimizations. This algorithm is also robust when inaccurate geo-location information is available.

4.2.1 Algorithm description

This algorithm shares the same assumptions as the ones listed in Section 4.1 except that we allow certain inaccuracy in the self-localization system of the sensors. Also, no range adjustable sensors are considered in this algorithm.

The algorithm involves a continuous moving process of all sensors periodically. We call each period a round of the algorithm. In each round, three main steps are described.

Setp 1: Location information gathering

In this step, each sensor will get the location information of its neighboring nodes. There are two ways to get the location information of other sensors: a sensor can directly measure other sensors' relative positions, or each sensor can self-locate and broadcast to other sensors. The latter is more applicable and is accounted for in assumption 4.

Setp 2: Region and environmental analysis

In this step, each sensor uses the location information to generate a local map and analyzes the region and environment by maintaining a region table. The region table is related to the ideal coverage model.

In addition to the ideal coverage model, in this algorithm, we divide the area near each sensor into six regions as shown in Figure 4.12. In each region, only the sensor closest to the central sensor is taken into consideration, such as sensors C, D or F. The objective of the algorithm is to relocate the sensors so that the distance between the closest sensor pairs (d_1, d_2, and d_3 correspondingly) is $\sqrt{3}R_S$.

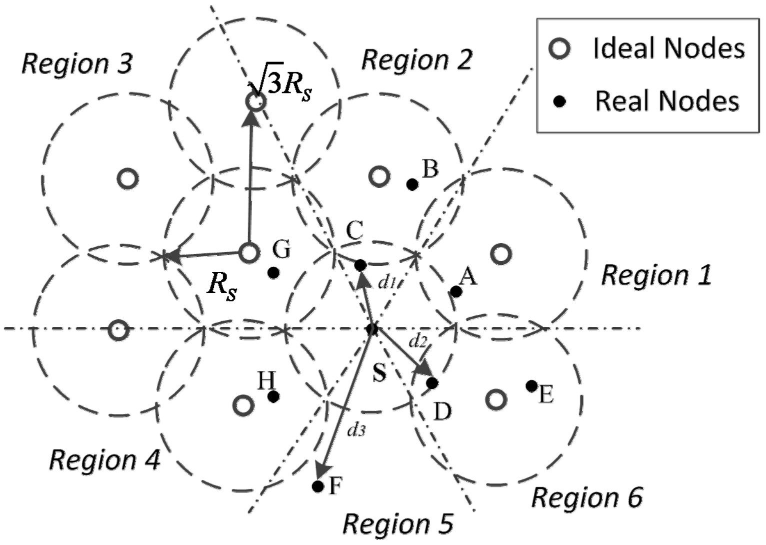


Figure 4.12 Ideal coverage model and region divisions.

The region table will include information of the closest neighboring sensor in each region, such as distance and relative direction. Also, a sensor will find out the shortest and longest distances of the sensors chosen, in all regions. The region table is of the form found in Table 4.2. The R_C value is used if there are no sensors in a certain region.

Table 4.2

REGION TABLE

Region Number	Sensor Number	Distance	Direction	Additional
1	A	4	30^o	
2	C	3	100^o	Shortest
3	G	5	135^o	
4	H	5	230^o	
5	F	7	260^o	Longest
6	D	3.5	320^o	

Step 3: Decision making and movement

We define the objective distance as a scalar:

$$d_0 = \sqrt{3}R_S \tag{4.15}$$

The two vectors d_s and d_l are defined as the vectors with the shortest and longest distance to the respective node from the central sensor node, in all six regions, (region 2 and 5 in Table 4.2):

$$\overrightarrow{d_s} = Amp_s \angle angle_s \tag{4.16}$$

$$\overrightarrow{d_l} = Amp_l \angle angle_l \tag{4.17}$$

As an example from Table 4.2, d_s will be a vector with $Amp_s = 3$ and $angle_s = 100^o$, and d_l will be a vector of $Amp_l = 7$ and $angle_l = 260^o$.

The direction and angle a sensor will move is set as:

$$\overrightarrow{M} = (Amp_s - d_o)\angle angle_s + (Amp_l - d_o)\angle angle_l \tag{4.18}$$

Additional: Boundary avoidance

Virtual nodes will be inserted into the algorithm to deal with boundary avoidance. Sensors treat the boundary as a mirror and create virtual nodes from the reflected images, which are included in Region Table calculations.

4.2.2 Simulation and Results

In our simulation, 20 sensors are deployed in a 100 meters by 100 meters square field. The sensing field is treated as a 100 by 100 grid when we calculate the coverage. The coverage ratio is fined as the ratio of the number of grid points that have a detection probability of 1, in binary model, or greater than 0.8, in probability model, to the whole coverage area.

a) Simulations using Binary sensing model

In this part, the sensing range is set to 17 m, which is sufficient for full coverage.

Three different scenarios are analyzed with random initial deployments. In scenario 1, sensors are deployed in the entire sensing field. In scenario 2, sensors are split into two groups near the left and right boundaries. In scenario 3, sensors are randomly deployed in the central 50m by 50m area of the sensing field.

Statistical methods are used to analyze algorithm performance. Specifically, 10,000 different initial deployments are applied for each scenario. Initial and final coverages are compared. Figure 4.13 indicates the probability density of the initial coverage ratio of all three scenarios. It can be seen that when all sensors are randomly deployed in the entire area in scenario 1, an average coverage around 80% can be achieved and when sensors are split as in scenario 2, 68% area can be covered on average; and there is only 47% coverage can be achieved on average in scenario 3.

Figure 4.14 shows the coverage percentage after 10 rounds of the self-relocation algorithm. It can be seen that in all the scenarios the algorithm performs very similar and has the same final coverage probability distribution. On average, 95.3% area can be covered after 10 rounds of the algorithm. More than 99% of initial deployments will result in a coverage ratio that is larger than 90%. These results prove the convergence of our algorithm.

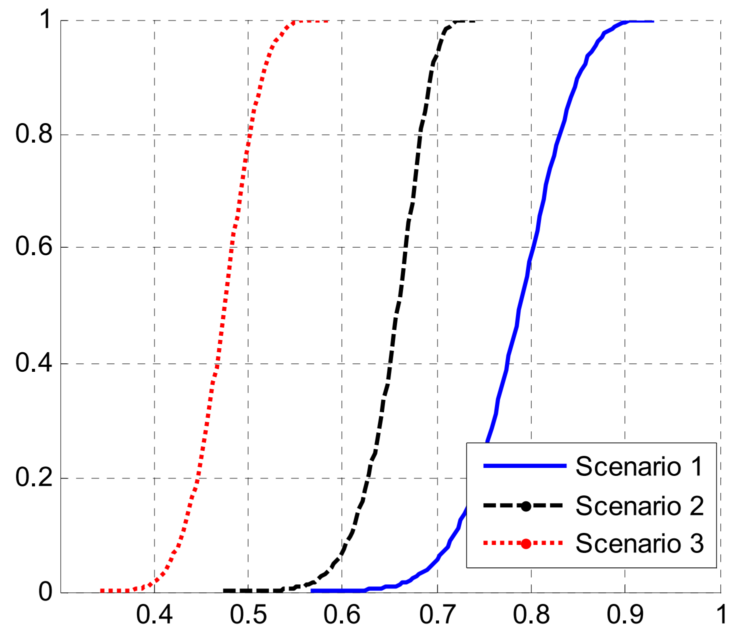


Figure 4.13 Probability of inital coverage ratio in three scenarios.

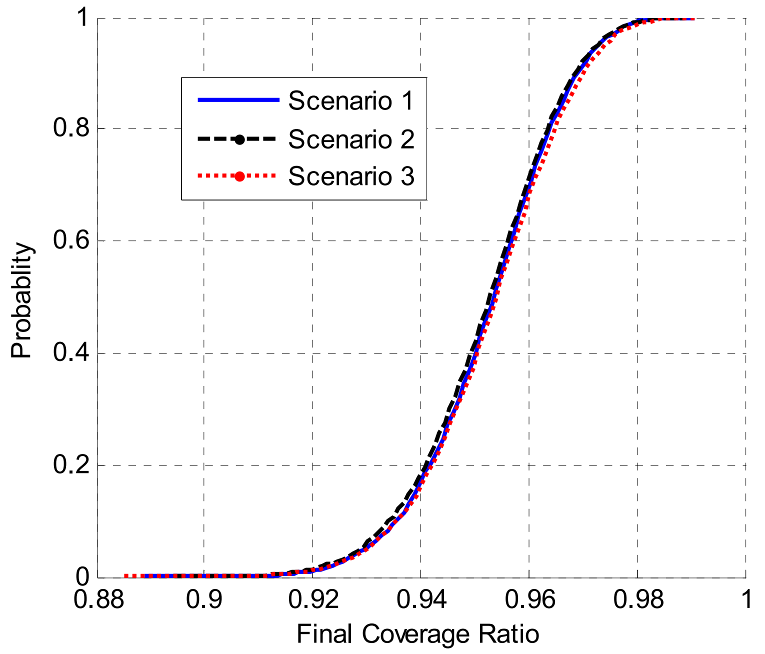


Figure 4.14 Probability of final coverage ratio in three scenarios.

Figure 4.15 plots the coverage ratio versus the round number of the algorithm. Here, three typical examples are taken randomly from the 10,000 samples in each case. It can be seen that in all three cases the algorithm converges within 10 rounds. This shows that 10 rounds are enough for the algorithm to converge and guarantee a better coverage than the initial deployment. In this aspect, the computation and reaction cost is low.

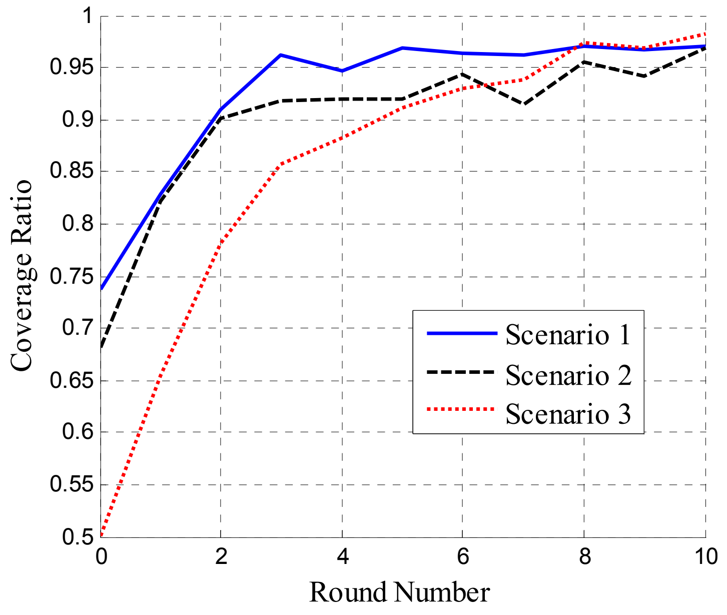


Figure 4.15 Examples in coverage ratio convergence for all scenarios.

Since our self-relocation algorithm converges for all three scenarios, in what follows we only proceed with simulations of scenario 3 for the probability sensing model.

b) Simulations using Probability sensing model

In this part, we use the probability sensing model with parameters as follow, R_S= 17 m, R_E = 3m, $\lambda = 2$, β =1. For these parameters, the sensors' sensing probabilities are continuously changing from 1 to 0.

The initial deployment is the same as scenario 3 of binary model, with 47.6% average initial coverage, which is reasonably higher than the binary model. Final coverage after our algorithm has run has a similar distribution to the result in binary model, but with a 95.5% average.

c) Simulations with inaccurate sensor locations

We also evaluate the performance of the algorithm when the locations of the sensors are not accurate. Noise is added to the location of the sensors in location gathering process. Therefore, each sensor will have certain inaccuracy in the estimation of its location. This is implemented in our simulation by adding uniformly distributed noise to both x and y coordinates of all sensors.

Three cases are examined: (a) case 1 with average noise of 0.5 m; (b) case 2 with average noise of 1 m and (c) case 3 with average noise of 2 m. Simulation results are shown in Figure 4.16. It can be seen that compared the noiseless case, 0.5 m and 1 m average noise has no big impact on the final distribution. Even when the noise has been increased to 2m on average, the average coverage ratio only changed from 95.3% to 94.7%. The results show that our self-deployment algorithm is robust against the existence of inaccurate sensor location data.

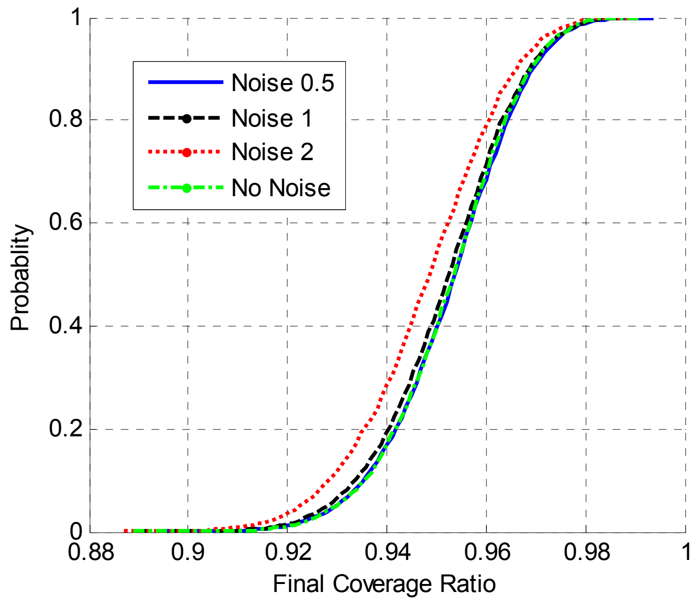


Figure 4.16 Probability of final coverage ratio with different noise in sensor locations.

4.2.3 Conclusions

This section introduces a distributed self-relocation algorithm in order to help sensor networks improve coverage using the mobility property of the sensor nodes. The algorithm can be used in different sensing models such as the binary model and probability model. The statistical results show the coverage can be increased to 95% on average even if error in the locations of sensors occur.

4.3 Optimization without Localization System

In some applications, such as indoor environments, localization schemes are not applicable or not accurate enough to help locate the sensors. Also, the cost and accuracy of GPS need to be considered. For example, the GPS kits sold by TI that has accuracy around 3 meters cost $40 each [88]. Therefore, we developed an optimization algorithm without using localization systems. Here, only the received signal strength between sensors is used for estimating the distance between them. This algorithm has the property of self-healing when some of the sensor die.

4.3.1 Assumptions

In addition to the general assumptions in Section 3.7, there are additional assumptions for this algorithm.

1. All sensors are range adjustable sensors.
2. All sensors know the approximate, total area *A* of the sensing field before being deployed. The sensors also know the approximate locations of the boundaries of the required sensing field.
3. There will be no localization system installed on the sensor nodes. They can only use received signal strength to obtain the distances between their neighbor sensors.
4. Obstacles may occur in the sensing field and they will follow the obstacle models described in Section 3.6.
5. No sensor will die in the early self-relocation time.
6. Only the binary sensing model is used in this algorithm to analyze the performance, for simplicity.

4.3.2 Algorithm Description

a) Algorithm Outline

Our aim is to design a distributed self-relocation algorithm for sensor networks that optimizes their coverage while using the least amount of energy. Our algorithm must also be able to perform self-healing when sensors die. Our algorithm relies only on the distance between sensors, which can be obtained by the received signal strength. In fact, no location or relative location information is used. Specifically, our algorithm has three parts: self-relocation, sensing range adjustment, and self-healing.

In the first step, sensors calculate the distance from other sensors by using the received signal strength of the “hello” information. Based on the distance information obtained, sensors move and relocate to spread themselves to optimal locations. In this step, sensors also need to avoid obstacles. After their relocation, sensors perform a sensing range adjustment. The sensors will not move in this step. If sensors die because of energy failure or are physically destroyed, a self-healing process is activated. Then, sensors recalculate the sensing range and relocate again.

b) Key concepts of the algorithm

There are some key concepts of our relocation algorithm. The objective of the algorithm is to achieve the ideal optimized deployment. In pursuit of this, there are calculation-related decision-making and destination-setting methods in the relocation process. Additionally, there are the methods that deal with sensing field boundaries and obstacles.

1) Ideal optimized deployment and threshold calculation

The threshold calculation is the same as in the optimization with Average Relative Position in Section 4.1.2. However, in this algorithm, locations and relative positions of sensors are unknown and the Voronoi diagrams are not applicable for deciding the sensing radius. Instead, we use the relationship between sensing radius and distance threshold to obtain the sensing radius each sensor should use:

$$R_S = \frac{d_{th}}{\sqrt{3}} \tag{4.19}$$

where d_{th} can be calculated with equation (4.2) and we can obtain the R_S values by using the sensing field area information (A_{Total}) and total sensor number (N):

$$R_S = \sqrt{\frac{2}{3\sqrt{3}} \frac{A_{Total}}{N}} \tag{4.20}$$

Also, considering the sensor nodes in the boundary areas, they will have smaller effective areas than other ones. The real threshold and sensing radius needs to be increased, so a 15% increase is used here.

2) ***Distance calculation between sensors***

The free space transmission propagation model, [89], is used in our deployment algorithm for obtaining the distance between sensors:

$$P_r = \frac{P_t G_t G_r}{\left(4\pi d / \lambda\right)^2} + N \tag{4.21}$$

where P_r is the received signal strength. P_t, is the transimitted power. G_t is the gain of transmitter, G_r is the gain of the receiver, and d is the distance between the transmitter and receiver. N is the noise component. Here, we consider the distance between the transmitter and receiver to be the distance between sensors.

Since the values of P_t, G_t, G_r, and λ here are fixed before the sensors' deployment, they can be preprogrammed into the sensors. Furthermore, the sensors can compute the distance between themselves and other sensors by using the equation below once they obtain the received signal strength and by neglecting noise:

$$d = \frac{\lambda}{4\pi} \sqrt{\frac{P_t G_t G_r}{P_r}} \tag{4.22}$$

3) ***Virtual Nodes***

For the sensors close to the boundary of sensing field, the algorithm generates virtual nodes along the boundary. Virtual nodes do not exist but their location information will be used in our calculations to prevent sensors from getting too close to the boundary. An example of virtual nodes is shown in Figure 4.17. Virtual nodes will be used in sensors' initial relocation and self-healing process.

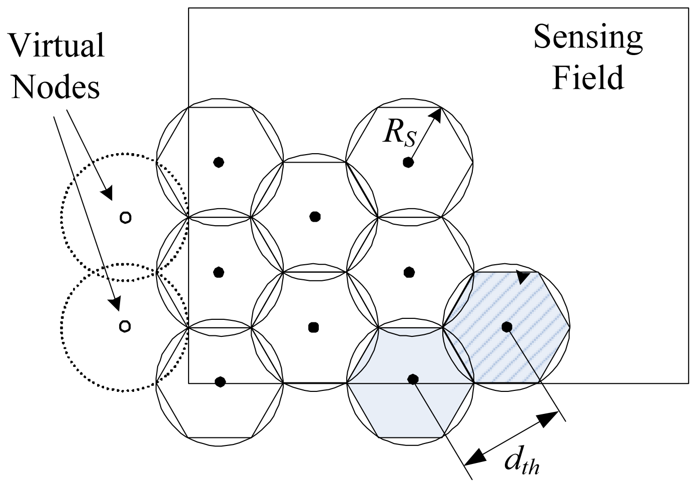


Figure 4.17 Virtual nodes in ideal optimized deployment.

4) ***Moving Criteria***

The goal of our relocation algorithm is to provide a deployment as close as possible to the ideal optimized distribution. It should be pointed out that sensors have no information about the direction of other sensors. Therefore the only information that should be used for relocation is the distance between the sensors. A sensor that needs to move is either too close or too far from other sensors. The moving criteria is described as follows:

Criterion 1: A sensor *S* needs to *move away* from other sensors if there is at least one sensor in its communication range of a distance less than $0.9d_{th}$;

Criterion 2: A sensor *S* needs to *move closer* to other sensors if criterion 1 is not met and no more than 2 sensors are at a distance from S that is less than $1.1d_{th}$;

Criterion 3: If criteria 1 or 2 are not met, a sensor does not need to move.

Here, a 10% margin is used so that sensors can achieve a distance close to d_{th} from other sensors.

5) ***Moving destination***

Since there are two criteria for each sensor, our algorithm will calculate the moving distance for each sensor based on the following equation:

$$d_{travel} = \begin{cases} d_{th} - \frac{1}{m_1}\sum_{j=1}^{m_1} d_j & \textit{for criterion } 1 \\ \frac{1}{m_2}\sum_{i=1}^{m_2} d_i - d_{th} & \textit{for criterion } 2 \\ 0 & \textit{for criterion } 3 \end{cases} \tag{4.23}$$

where d_j and d_i are the distances between sensor *S* and other sensors, and in Criterion 1 only the sensors that are closer than the distance threshold d_{th} are counted, and the total number of such sensors is m_1; in Case 2 all the neighbors that sensor *S* knows are counted and the total number of those sensors is m_2.

The relative locations of neighboring sensors are unknown. Therefore, our algorithm chooses randomly the direction to move. In order to avoid sensors from moving back and forth, a direction control scheme is used. The change of direction between each movement will be less than 90 degrees. Each sensor records the direction α it used in its last movement, and randomly picks another direction within the range of α-90^o to $\alpha+90^o$, for the direction of its next movement. The goal of movement for criterion

1 is to increase the average distance between sensors, and the goal for criterion 2 is to decrease the average distance between sensors. Therefore, after sensors randomly choose direction, they will check if going to those directions will meet the goals by moving a short distance to the chosen direction and checking the received signal again. If the random movement does not produce the desired outcome, the sensor moves back and stays at the same position in that round.

6) ***Self-healing process***

We assume that a sensor has n_1 neighbors it can communicate with after the relocation is finished. After the WSN operates for some time, some sensors lose their functionality and stop working. The sensor now has only n_2 neighbors. It will recalculate the distance threshold and sensing range, and then follow the relocation procedure to adjust their locations using the following:

$$d_{th2} = \sqrt{\frac{n_1+1}{n_2+1}} \cdot d_{th1} \tag{4.24}$$

$$R_{S2} = \frac{d_{th2}}{\sqrt{3}} = \sqrt{\frac{n_1+1}{n_2+1}} \cdot R_{S1} \tag{4.25}$$

where all subscripts “1” refer to the value before self-healing process starts, and the subscripts “2” refer to the new calculated value.

7) ***Obstacle avoidance***

When an obstacle blocks the sensors’ relocation path, the sensor has to stop before it hits the obstacle. In the next round the sensor plans its route to move around the obstacle, it can choose to go clockwise or counterclockwise, which depends on which direction satisfies the moving destination calculation. For program simplicity, only regularly shaped (rectangular) obstacles are considered.

c) Relocation Process

The detailed process of the relocation scheme is as follows:

Step 1: Pre-knowledge installation

- Load sensing range, sensing area and sensors number to sensors memory.
- Load desired round number, set the current round to be round 1
- Calculate d_{th}, r.

Step 2: Round Initialization

- Broadcast and receive “hello” information;
- Calculate the average distance d between sensors by the receiving signal strength;
- Compare the distance condition with the criteria condition and make a moving decision
- Calculate d_{travel} if needed
- Broadcast moving decision;
- Receive moving decision from other sensors
- Record the number of neighboring sensors that need to move

Step 3: Self-relocation

- If no other sensor nodes declared moving direction choosing process in Set Random time T (conflict avoidance)$\rightarrow$
  - ➢ Broadcast “Self-relocation ON”

- ➢ Choose random direction and move short distance
- ➢ Broadcast message “Reference node”;
- ➢ Receive “hello” message respond to message “Reference node”;
- ➢ Re-calculate average distance *d*’;
- ➢ Decide moving direction according to the criteria objective;
- ➢ Move to destination position and send message “Self-relocation OFF”;

- If a “Self-relocation ON” is received, a sensor will not start the timer T. It will wait until a message “Self-relocation OFF” is received.
- When all sensors finish relocating, this round of the algorithm is finished.
  - ➢ If the round number is equal to maximum allowed round number, go to step 4.
  - ➢ Otherwise, go to step 2 and start a new relocation round

Step 4: All sensors stop and adjust sensing range to *r*

Step 5: Self-healing

- If a sensor detects loss of other sensors, perform self-healing process.

4.3.3 Simulation and Results analysis

In this simulation, a 100 m by 100 m square sensing field is used. The minimum distance between grid points is 1 m. In this example, we use a sensor network for surveillance that is comprised of motion detector sensors. The sensing range is usually between 18 – 25 m. Therefore, we set the maximum sensing range for our algorithm to be 25 m. Also, the communication range is set to be 55 m, i.e., two times larger than the sensing range in order to guarantee network connectivity. This is also a practical communication range since, for example the zigbee communication range is 75 m. In this case we did not set the communication range as 75 m because the sensing field is only 100 by 100 m.

a) Self-relocation algorithm performance in non-obstacle environment

In the first part, 20 sensors are deployed in the sensing field. Assuming a 15% increase as explained in Section 4.3.2, the distance threshold and sensing range can be calculated by equation (4.2) and (4.20):

$$d_{th} = 1.15 \cdot \sqrt{\frac{2}{\sqrt{3}}\frac{A}{N}} = 1.15\sqrt{\frac{2}{\sqrt{3}}\frac{10^4}{20}} = 27.6m \tag{4.26}$$

$$R_S = \frac{d_{th}}{\sqrt{3}} = \frac{27.6}{\sqrt{3}} = 15.9m \tag{4.27}$$

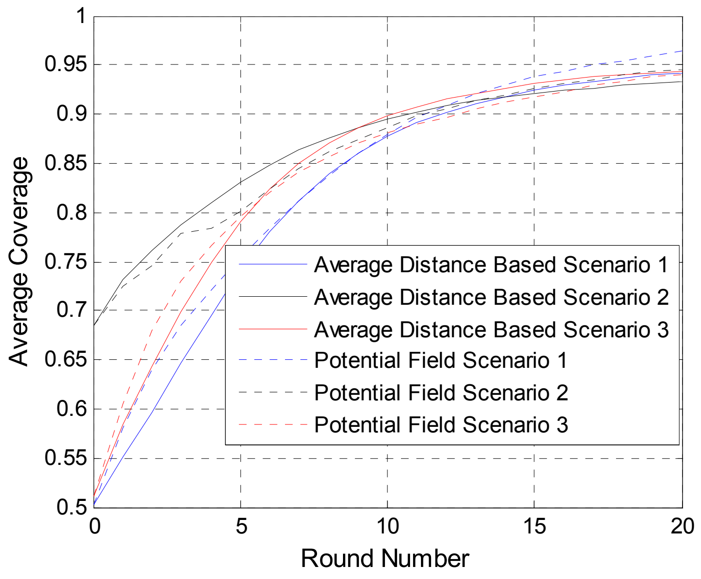
Three scenarios are simulated, in which all sensors are randomly deployed initially. In the first scenario, all sensors are deployed in the central 50 by 50 meters area; in the second scenario all the sensors are randomly deployed in the entire area; and in the third scenario, sensors are split into two groups and deployed near the left and right boundary. The initial coverage of the three scenarios are 50%, 68% and 51%, respectively, when we use R_S*=15.9 m*.

- ***Coverage Analysis***

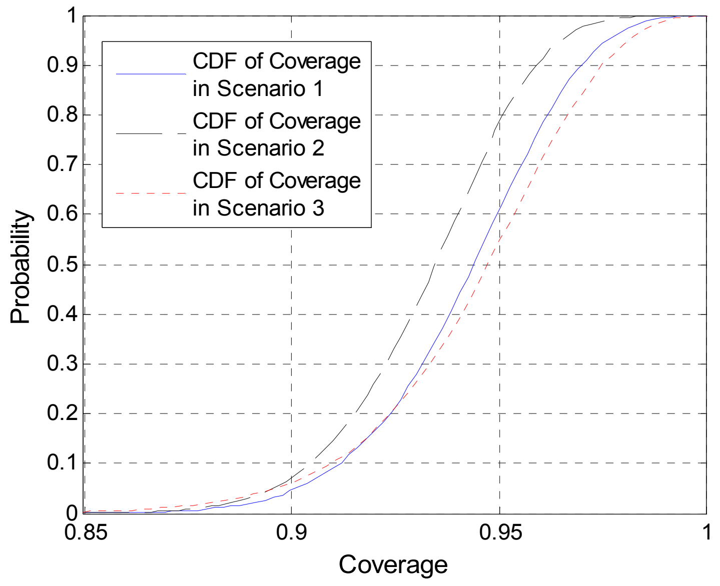
Since the algorithm is based on a random selection of directions, a statistical method is used for analyzing the results. Each scenario is run 10,000 times and the desired round number is 20 for all the scenarios. Results are shown in Figure 4.18.

Figure 4.18(a) shows how the average coverage increases as the relocation process runs. Specifically, on average 88% coverage is achieved after 10 rounds. Also, the coverage converges to approximately 94% in all scenarios. We also apply the potential field method with the parameters described in [8] in order to compare it with our algorithm. The potential field algorithm provides a converged coverage of 95% after 20 rounds. Therefore, our algorithm and the potential field algorithm converge to approximately the same coverage. Also, our algorithm and the potential field algorithm exhibit the same convergence rate. However, our algorithm does not require the use of GPS or any other self-localization hardware that increase the complexity and cost of the sensor nodes. Also, our algorithm can be used in environments where GPS does not work, such as, indoor, underground, underwater, etc.

Figure 4.18 shows the Cumulative Distribution Function (CDF) for the coverage achieved after 20 rounds. It is seen that the final coverage after 20 rounds is between 85% and 100% for all three scenarios. It can also be seen that the probability of achieving at least 90% coverage is larger than 90%. Finally, the standard deviation of the final coverage after 20 rounds for scenarios 1 and 2 is 2.2%, and for scenario 3 is 2.6%.



(a) Average coverage vs round number



(b) Coverage after 20 rounds in statistic probability

Figure 4.18 Simulation results for self-relocation algorithm.

- ***Energy analysis***

Energy models described in Section 3.4 are applied here for our energy analysis. We assume the sensors in the algorithm move in a constant speed and also turn in constant speed. Therefor, the energy consumed by a sensor while moving and turning is linearly related to the distance it travels and the direction change in angle. Specifically, in each round, the energy consumption is cacluating using the following two parts:

Travel*:* the energy consumed by movement in one direction for each node in each round is calculated by:

$$E_{dis} = d_{travel} \cdot 9.34\,(Joules) \tag{4.28}$$

Direction Change*:* In step 3 in our relocation algorithm, a sensor needs to move a short distance and recalculate the signal strength. After this calculation, a sensor needs to decide to follow that direction or go back to the original position. The energy consumed by the turning process can be calculated by:

$$E_{Turn} = \begin{cases} (A_{diff} / 90) * 2.35 \, (Joules) & Keep\ Moving \\ (360 / 90) * 2.35 \, (Joules) & Turning\ Back \end{cases} \tag{4.29}$$

where A_{diff} is the direction difference of the direction before and after the step. If the direction does not meet the requirement for signal strength, the sensor node has to turn back to the original position and the original direction. This is equivalent to turning 360 degrees.

The energy consumed by Potential Field algorithm is also calculated as a comparison. In additional to the energy consumed by travel and direction changing, energy consumed by GPS chip is also included. According to [16], the GPS chip consumes 198 mW. Assuming that sensors move at a speed of 0.08 m/s and neglect the time in between each round for sensors to stop and calculate, so that the GPS consumes 0.198*(1/0.08) = 2.475 Joules per meter.

Figure 4.19 shows the average energy consumption of sensors versus the coverage percentage for our algorithm and the potential field method. It can be seen that the potential field algorithm uses less energy compared to our algorithm. The provided savings in energy consumption are around 25% to 30% depending on the initial deployment. This happens because in the potential field algorithm every sensor knows the location or relative location of all other sensors thereby making better decisions for the directions that the sensors need to move toward. On the contrary, our algorithm consumes more energy when it turns back and forth. However, our algorithm does not require that sensors know others location or transmit their location information to other sensors. This can provide significant savings in terms of hardware cost since no GPS hardware is required.

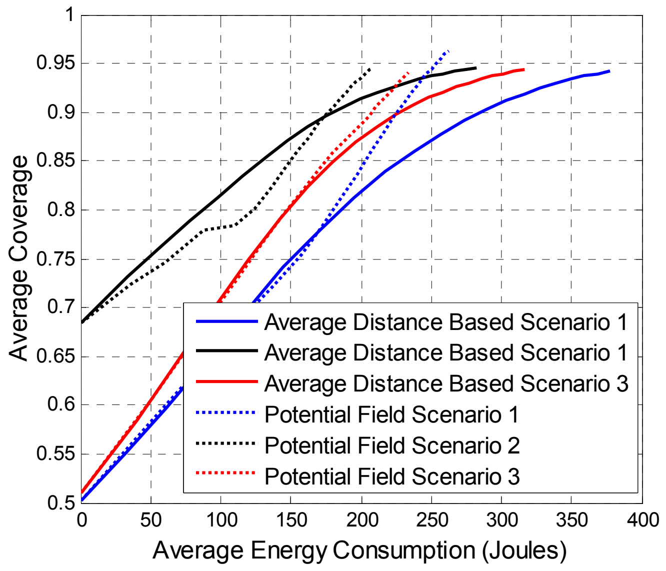


Figure 4.19 Simulation results for average coverage and energy consumption.

b) Simulation for self-healing algorithm

After the execution of the self-relocation algorithm, we randomly choose five sensors to die. The initial deployment for our self-healing simulation is shown in Figure 4.20. The 5 nodes without coverage are assumed dead.

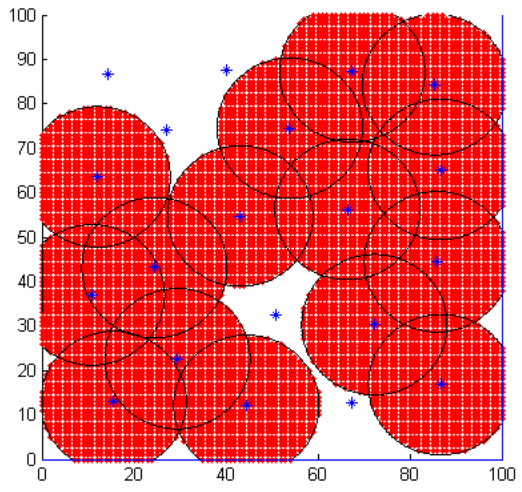


Figure 4.20 Five sensors die out of 20 originally deployed sensors.

The simulation results are shown in Figure 4.21. This simulation is also run 10000 times. Figure 4.21 shows the average, best case and worst case coverage. After the sensors have died, only 80% coverage is provided. However, our algorithm increases the coverage by increasing the sensing range of each sensor. After the first round, the coverage ratio increases slowly because the sensors need to move slowly toward the areas that are not covered. After 20 rounds, our algorithm achieves an average coverage of 93%. In this case, the average sensing range of the sensors is approximately 18.3 m thereby providing 26.8% energy savings compared to the case where the sensors use their maximum sensing range of 25 m. The average traveled distance in the self-healing process is around 7.5 m.

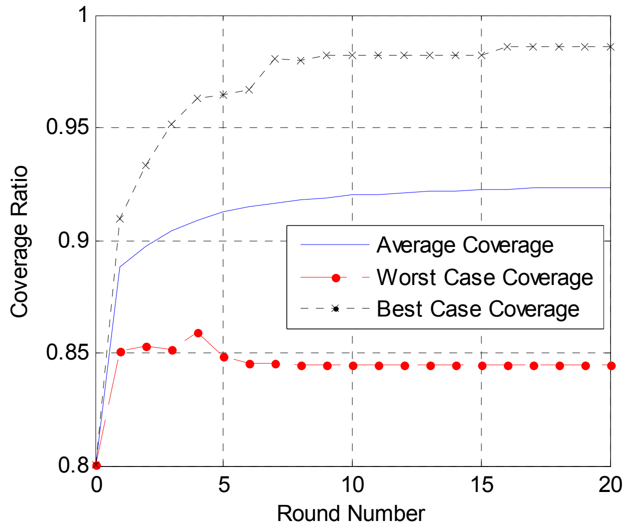


Figure 4.21 Coverage using the for self-healing algorithm.

Figure 4.22 shows one sample simulation for our self-healing algorithm. The movement of all the sensors are shown. Blue dots are the initial position of the sensors, and the red stars are final positions. Figure 4.22 also shows the final uncovered area in red. It can be seen that in the left up corner where three sensors died, other sensors have not move enough to completely cover this area. However, for the other two places, where only one sensor has died (middle and bottom of the sensing field), the self-healing algorithm provides almost complete coverage of the area that the dead sensors used to cover.

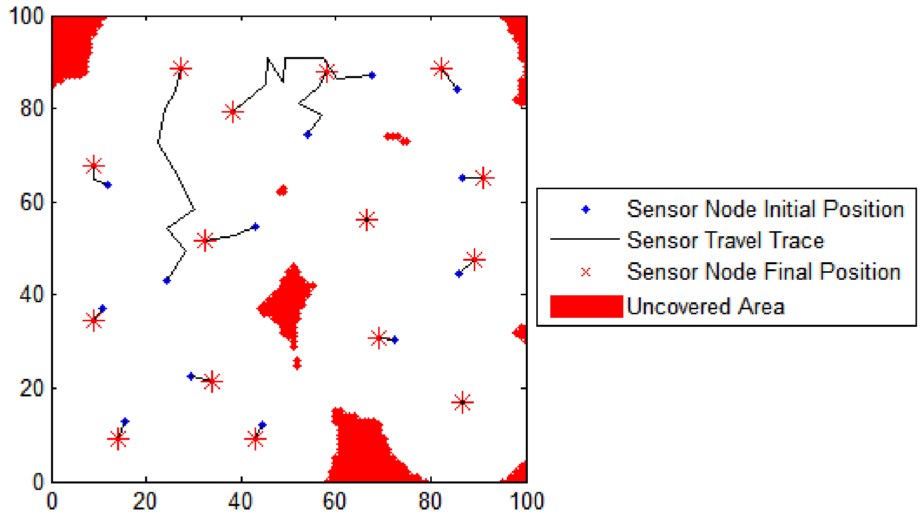


Figure 4.22 A sample simulation of self-healing process.

c) Self-relocation algorithm performance in environment with obstacles

In this part, two obstacles are placed in the sensing field and 20 sensors are in the middle part of the sensing field which is shown in Figure 4.23. Due to the appearance of obstacle, 10% increase in sensing radius is applied. The initial coverage is 19% with the sensing radius of 17.5 meters.

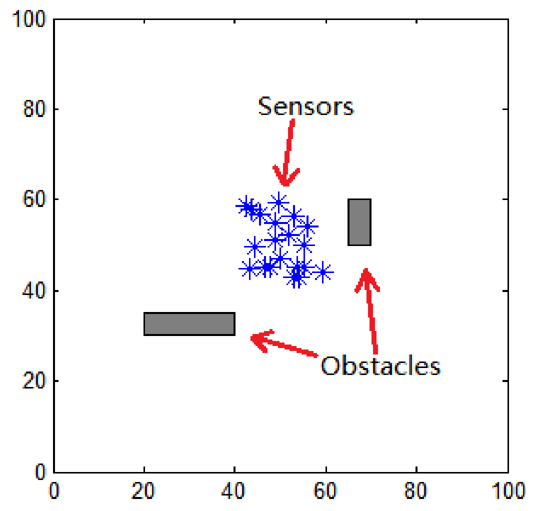


Figure 4.23 Initial deployment in field with two obstacles.

Figure 4.24 shows the CDF of the final coverage after 20 rounds. In fact, 90% coverage is achieved on average over 20 rounds. The standard deviation of the final coverage is 3%. Since the obstacles occupy 150 m^2 in total, the maximum coverage cannot exceed 98.5%.

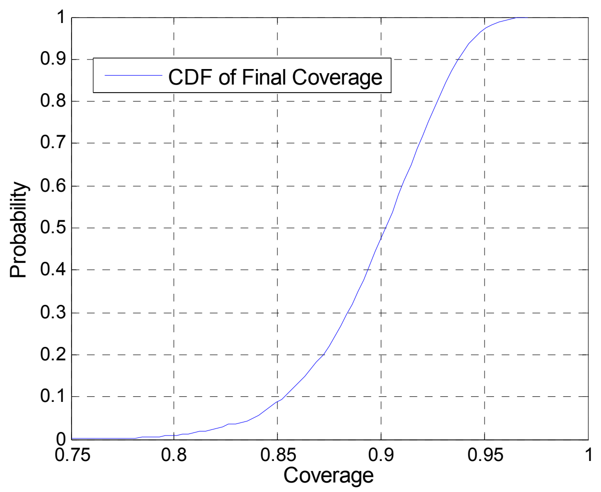


Figure 4.24 Statistical coverage result in self-relocation algorithm with obstacles.

4.3.4 Conclusions

In this section, we developed a self-relocation and self-healing algorithm based on average distance. The sensors only need to know the approximate area of the sensing field and the total number of sensors before they are deployed. The sensing range of each sensor is adjusted to reduce the redundancy and provide energy savings. No geo-location or relative location knowledge is needed thereby providing significant savings for each node since the GPS hardware is not required. However, the tradeoff is that the sensor nodes have to consume more energy to achieve a certain percentage of coverage compared to the algorithms using GPS devices. The algorithm works in both environments with and without the appearance of obstacles. The disadvantage of this algorithm is that the direction movement is random and not always the best direction thereby resulting in a larger travelled distance. The algorithm is suitable for emergency surveillance where sensors do not have GPS devices. Therefore, this algorithm is preferred when sensor nodes can localize themselves by cooperating with other sensors, only in emergencies or when triggered events occur in their sensing range.

4.4 Summary

Three distributed optimization algorithms were introduced for relocating the sensors in order to optimize sensing coverage. In these algorithms, sensors work cooperatively with each other.

Optimization with relative-position algorithms are developed for sensors networks that have geo-location devices. Instead of incrementally moving each sensor every round, the algorithm optimizes the average relative distance so as to reach the ideal optimum layout as soon as possible. The algorithm estimates the destination distances using the global information, and then it separately considers the sensors that should move apart from each other and those that should get closer to avoid uncovered areas between them. This enables the algorithm to converge in approximately 5 rounds after the sensors are deployed. The trade-off for the fast convergence is that the optimized sensing range will be 1-2% less than the one obtained by existing algorithms.

Optimization with weighted, relative-distance algorithms is designed for the sensors networks with inaccurate localization systems. The algorithm separates each sensors' neighboring area into six subareas and calculates the most weighted relative distances. Two of the most weighted relative distances among in the six subareas are used. Therefore, the error in the localization systems from the other four subareas is eliminated so that the algorithm can perform well against the total error due to the collective localization systems of these sensors. As a tradeoff, 3% less coverage can be achieved compared to existing solutions.

The algorithm that does not use the sensors' locations can also provide optimized coverage, but with increased energy consumption as a trade-off. It uses the randomly chosen directions to decide its relocation destination. As a consequence of not optimizing the direction, a greater total travelled distance of all the sensors is required. Compared to the potential field algorithm, when optimal layout is achieved, sensors consume 25% to 30% more energy in movement. However, our algorithm is the first approach in coverage optimization without sensor location information. It is particularly useful for indoor applications for which self-localization systems cannot provide accurate location information.

CHAPTER 5

CENTRALIZED OPTIMIZATION ALGORITHM DESIGN

This chapter details two centralized algorithms for optimizing the coverage of wireless sensor networks. The first algorithm is based on a Genetic Algorithm, and the second algorithm is based on Particle Swarm Optimization. Both of the algorithms require large amounts of computation, so they cannot be performed by regular sensor nodes. Instead, these algorithms are only possible when powerful central nodes are available. For both algorithms, theoretical analysis and simulations are performed.

5.1 Multi-Objective Genetic Algorithm

In this section, we propose a Multi-Objective Genetic Algorithm (MOGA) based on the sensor relocation algorithm. In this algorithm, we especially focus on the optimal coverage and minimum distance travelled for sensors, after their initial, random deployment. The two objectives of coverage and distance travelled by sensors are always in conflict. Therefore, trade-offs need to be made between the two. The fitness function is designed with respect to the different objectives, and the genetic algorithm is used to optimize the fitness function to deliver the best tradeoffs.

5.1.1 Assumptions for MOGA

In additional to the general assumptions in Section 3.7, there are more assumptions for this MOGA.

1. A powerful processor is needed. The algorithm will be performed by the processor. The processor can be installed in the base station.

2. The base station needs to know the location of each sensor after the sensors have been deployed. The base station also knows the map of the entire sensing field, such as the limits of the boundaries and obstacles. Obstacles in sensing field will not degrade the performance of the algorithm.
3. All sensor nodes can communicate to the base station directly or by multi-hopping. In this way, the base station can instruct the sensors to move as determined by the algorithm.
4. All sensors have a fixed sensing radius. No range adjustable sensors are required.
5. For simplicity, only the binary sensing model is used in these simulations of the algorithm.

5.1.2 Algorithm Outline

The objective of the MOGA is to relocate the sensors after their random deployment, with the aim of improving coverage. The main steps of the algorithm are described below:

1. Randomly deploy the sensors.
2. The base station collects the location information of all sensors. Sensors can perform self-localization and send their location information to the base station.
3. The base station uses the location and the pre-installed map information to design the fitness function needed to be optimized.
4. The base station performs the MOGA to determine the destination of the sensors' relocation position and sends the result to them.
5. All sensors relocate according to the instructions given by the base station.

It can be seen that the genetic algorithm will only be performed once and all sensors will only need to move once after they are deployed. However, if some sensors die, the sensing field coverage will decrease. In this situation, the algorithm can run again with the surviving sensor nodes and perform a self-healing process to achieve the desired maximum coverage.

5.1.3 Algorithm objective and fitness function design

There are two main objectives of the MOGA. The first objective of our algorithm is to maximize the coverage ratio. The coverage ratio ($R_{Coverage}$) is defined in Section 3.2 equation (3.5). The second objective is to reduce the sensor's use of energy for mobilization, which is defined in Section 3.4. The energy for turning is negligible since the sensor only move once after initial randomly deployed. Therefore, in order to maximize the energy savings for the relocation of sensors we must minimize the travelled distance.

Here, the average travelled distance is used. Since the sensors are only going to move once, equation (3.20) for calculating the travelled distance can be simplified. Assume there are N sensor nodes in the sensor network, the initial and final locations of the sensing nodes are (x_i, y_i) and (x_i', y_i') correspondingly. Then, the average travelled distance is equal to:

$$\overline{D} = \frac{\sum_{i=1}^{N} D_i}{N} = \frac{\sum_{i=1}^{N} \sqrt{(x_i - x_i')^2 + (y_i - y_i')^2}}{N} \tag{5.1}$$

where D_i is the travelled distance of one sensor obtained by equation (3.20)

If an obstacle appears in between the initial and final locations of a sensor, the travelled distance will not be the direct distance calculated by equation (5.1). Instead, it has to be calculated segmentally as shown in Figure 5.1.

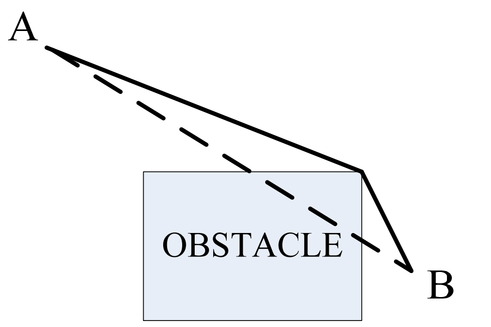


Figure 5.1 Distance calculation when an obstacle appears.

In our algorithm, the fitness function is designed according to our objectives. Our genetic algorithm is designed to seek the minimum value of the fitness function. Therefore, the coverage ratio, which we need to maximize, should be changed to a function that needs to be minimized. Specifically, we use here the uncovered area ratio:

$$R_{UNCOV} = 1 - R_{Coverage} \tag{5.2}$$

where $R_{Coverage}$ is obtained from equation (3.5).

The aim of our genetic algorithm is to minimize both the average travelled distance and the uncovered area ratio. It should be noted that there is a trade-off between the two objectives. Once the N sensors are deployed, the sensors remain at their positions, before the algorithm is executed. The larger the coverage needed, the longer the movement of the sensors will be. Conversely, such long movements of the sensors do not guarantee larger coverage. By using a multi-objective genetic algorithm, the best tradeoff between the coverage and the travelled distance can be achieved.

The fitness function of our problem is written as:

$$F = R_{UNCOV} + w \times \overline{D} \tag{5.3}$$

where w is the weight of the average travelled distance. The selection of w is related to the size of the sensing field and the number of sensors. The value of w is to balance R_{UNCOV} and $w \times \overline{D}$ at similar orders of magnitude. Our results will demonstrate that the chosen value of w controls the tradeoff between the two outcomes.

5.1.4 Algorithm description

Genetic algorithms are heuristic search algorithms that come from the idea of natural evolution. They are suitable for solving nonlinear optimization problems and for finding the global optimization value of fitness functions. Since the initial population covers only a group, local optimized values will be avoided.

In optimization problems with multiple conflicting-objectives, such as, minimizing energy and maximizing coverage ratio, the optimized fitness function will provide the Pareto optimal front. The Pareto optimal front represents the optimal trade-off between objectives. There will be no better fitness values unless at least one objective gets worse. In our case, this means if the optimized fitness function indicates that the average travelled distance of m_1 meters will result in R_1 for the coverage ratio, there will be no average travelled distance less than m_1 meters that will result in a coverage ratio larger than R_1. Also, (m_1, R_1) will be on the Pareto optimal front.

Our algorithm uses the genetic algorithm to minimize the fitness function value. In our problem the variables that are needed to be optimized are the final positions of the sensors. Therefore, the input variables to the genetic algorithm are the coordinates of the sensor nodes:

$$C=[x_1',x_2',\cdots,x_N',y_1',y_2',\cdots,y_N'] \tag{5.4}$$

which is a vector of length 2*N*. Different final positions of the sensors will result in different coverage ratios and travelled distances.

Detailed process of genetic algorithm is described as follow:

1. In the first generation, a certain number M (defined as population size) of vectors $C$are randomly generated. Since C is composed of the coordinates of sensors, the values of the components are limited by the boundary of the sensing field.
2. In each following generation, the coordinates x' and y' are exchanged and may also mutate with a certain percentage (i.e., the mutation ratio). If the ratio is too small, the optimal value may not be achieved, and if the ratio is too large, good 'chromosomes' of one generation may be changed, thereby making it hard to find the optimal value. For each generation, the vectors of M with the best fitness values will be kept, and the others will be discarded.
3. The algorithm will come to an end when either of the following two conditions are met: a) the value of optimum fitness function does not change between generations; b) the generation number has achieved the desired maximum generation number.

5.1.5 Simulations and results

In all our simulations, the sensing field is a 100 m by 100 m square. All the sensors are assumed to have the same sensing range of R_s= 25 m. The sensors are randomly deployed before the simulation starts. The population size for our genetic algorithm is 100 and the mutation ratio is 1%.

a) Sensing field without obstacles

There are three cases without any obstacles, where 10, 9 and 8 sensors are used. It should be pointed out that 9 sensors are sufficient to provide full coverage. Therefore, for the 10 sensors' case, a redundant sensor exists. Also, for the 8 sensors' case, full coverage cannot be achieved by any spatial distribution of the sensors. The initial deployment of 10 sensors is shown in Figure 5.2.

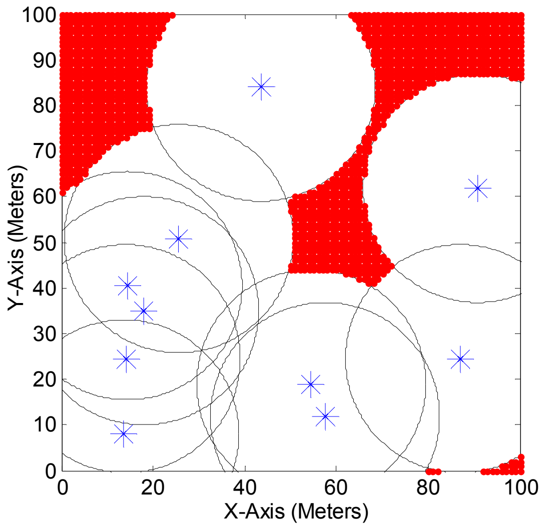


Figure 5.2 Initial position of 10 randomly deployed sensors.

Figure 5.3 shows the simulation results of our algorithm. Specifically, Figure 5.3 illustrates the trade-off between the two objectives, that is, the uncovered area ratio and the average travelled distance. When the travelled distance increases, the uncovered area decreases as expected. This outcome is an example of the Pareto optimal front. All the points above these curves are not optimal outcomes, because when we consider the same travelled distance, the maximum coverage that can be achieved lies on the curves of Figure 5.3.

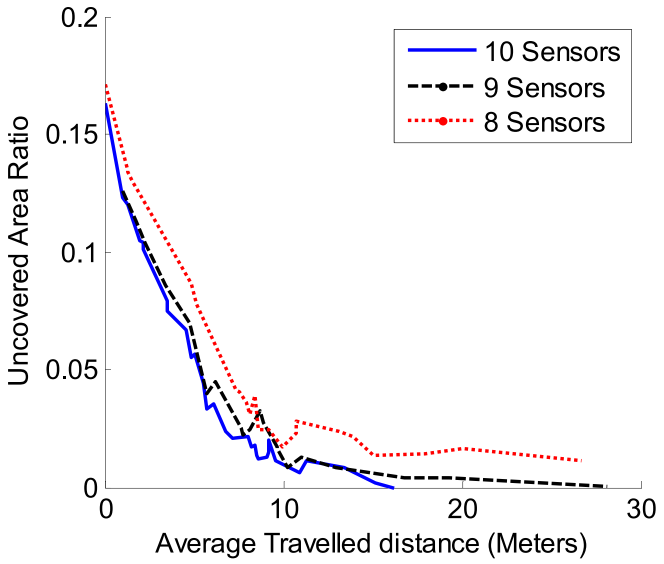


Figure 5.3 Relation between uncovered area and the average travelled distance.

Figure 5.3 also shows the area coverage versus distance travelled by sensors. When the travelled distance is equal to 0, (i.e., sensors are in their initial positions), 10 sensors and 9 sensors provide a coverage of approximately 84%, and 8 sensors provide coverage of 82.8%. The coverage increases as the average travelled distance by the sensors increases. Figure 5.3 shows that for the same travelled distance, the case of 10 sensors always provides the best coverage. This is expected, since the more sensors we use, the higher the coverage we can achieve. Also, Figure 5.3 illustrates that when the travelled distance is less than 10 m, the coverage increases much faster as the travelled distance increases. This empirical observation indicates that limiting the sensor average movement to 10 m achieves the best tradeoff between coverage and travelled distance (i.e., energy consumption).

Two Pareto optimal layouts for ten sensors are shown in Figure 5.4. Case A has an average travelled distance of 7.9 m and the coverage is about 97.6%, versus case B that has an average travelled distance of 16 m and 100% coverage. The shaded areas in Figure 5.4 correspond to uncovered areas, and the stars correspond to the locations of the sensors.

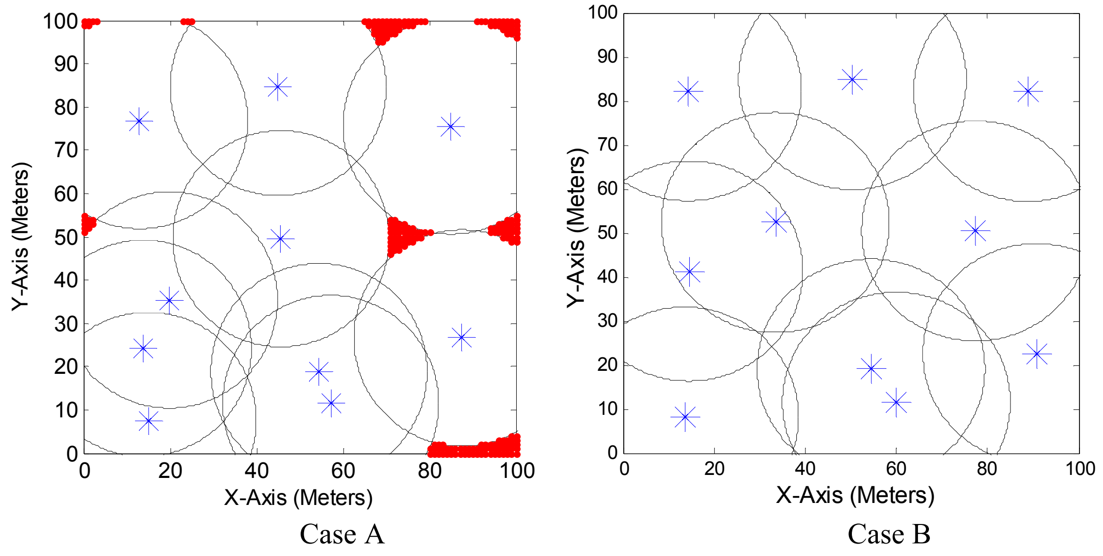


Figure 5.4 Different Pareto optimal layouts.

Furthermore, Figure 5.5 illustrates the relationship between the weight coefficient w and the optimized travelled distance, which shows that when w increases, the sensors tend to move less.

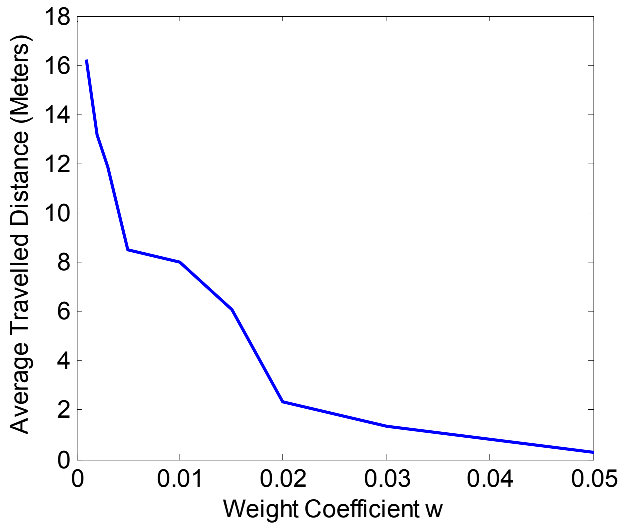


Figure 5.5 Average travelled distance versus the weight coefficient *w*.

b) Sensing field with obstacles

In our last simulation case, we incorporate the presence of obstacles in the sensing field. Specifically, we add an obstacle with the dimensions of 0.3 m by 6 m. If the genetic algorithm shows that the path between initial and final positions of the sensor is blocked by an obstacle, the travelled distance of that sensor will be calculated as the shortest path required to go around the obstacle. In this case, 8 sensors are initially deployed and the initial coverage is 72%. This initial layout is shown in Figure 5.6.

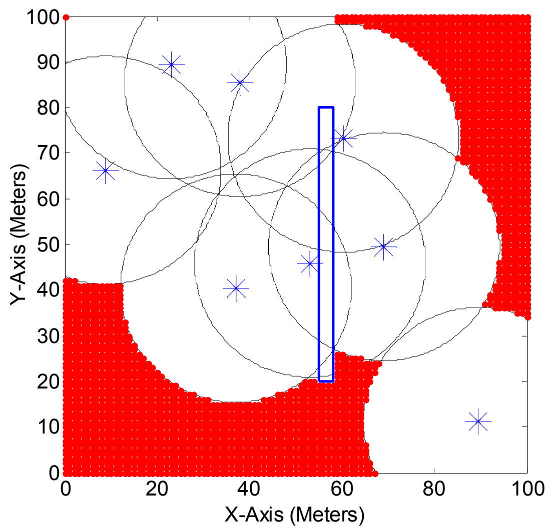


Figure 5.6 Random deployment of 8 sensors along with the coverage.

Figure 5.7 shows the simulation result of this case. As we can see, with about 5 m average travelled distance, the coverage can be increased from 72% to 91%. Further improvements of coverage require significantly larger travelled distance.

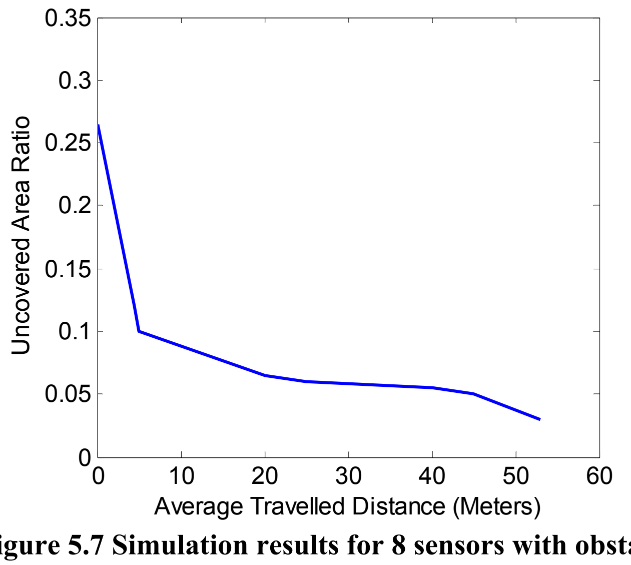
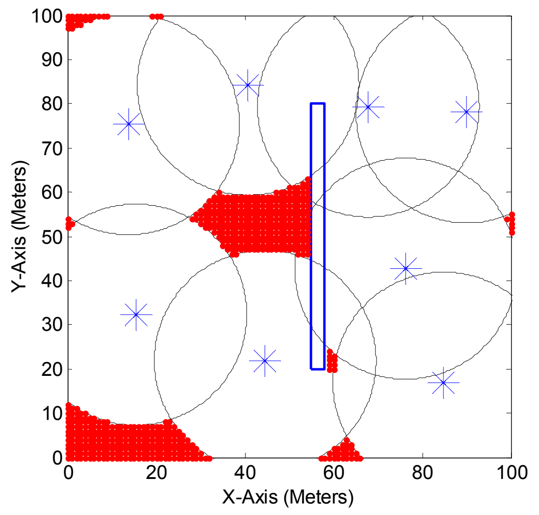
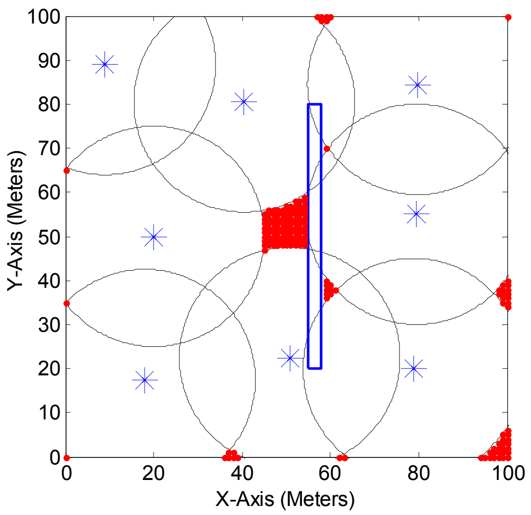


Figure 5.7 Simulation results for 8 sensors with obstacle.

Figure 5.8 shows two Pareto optimal layouts. Case A has a coverage of 92% and a travelled distance of 4 m, while case B has a coverage of 97% and a travelled distance of 53 m.



Case A



Case B

Figure 5.8 Different Pareto optimal layouts.

5.1.6 Conclusions

This section introduces a genetic algorithm for relocating nodes of a WSN in order to provide the trade-off between coverage and the travelled distance by the sensros. Both the coverage and the energy conservation are considered in our optimization. For each different layout, the shortest travelled distance route is found. Also, for a fixed travelled distance of all sensors, the coverage can get optimized using our algorithm. The results show that best coverage can be achieved by reducing the weight between travelled distance and coverage. The larger the weight on travelled distance is, the smaller the average travelled distance of the sensors is; this also provides reduced coverage. Also, our algorithm can simulate obstacles and provide the optimum distribution of sensors. Furthermore, the algorithm can be used in dynamic environments. It can also perform applied to sensor network healing when sensors die.

5.2 Particle Swarm Optimization

In this section, we propose a Particle Swarm Optimization (PSO) based on a centralized algorithm to relocate sensors after random deployment. Similar to the previous algorithm, this algorithm also aims at providing optimized coverage and energy consumption. The main difference between this algorithm and the one presented before is that Voronoi diagrams are used to ensure sensors with adjustable range can cover the entire sensing field. Thus, the problem is how to use energy in a manner that prolongs the lifetime of the sensor network. Both the sensing radius and travelled distance are optimized to save energy.

5.2.1 Assumptions for PSO

In addition to the general assumptions in Section 3.7, there are more assumptions for our PSO.

1. As the PSO is also a centralized algorithm, similar to the GA based algorithm in Section 5.1, it also needs a powerful processor in a central node to perform the algorithm. Also, the central node needs to know the map of the sensing field and the location of sensors. The sensors can communicate with the central node.
2. No obstacles are present in the entire sensing field. This assumption is made because the Voronoi diagram is used for getting the minimum sensing radius. If an obstacle appears, this method is not applicable.
3. All sensors are range adjustable sensors that can adjust their range as instructed by the central node.
4. Both energy models for adjustable range sensors are considered in this algorithm.

5. Only the binary sensing model can be applied in this algorithm.
6. It is assumed that the sensor network is functioning correctly only when all sensors are operating properly. If any sensors die, the central node has to restart the optimization algorithm to optimize the coverage and maximize the sensor network lifetime.

5.2.2 Optimization goals

There are two goals in our algorithm. First, we aim to provide or guarantee full coverage of the sensor network. Second, we aim to reduce the energy consumption of the sensors so as to extend the life of the sensor network.

The first goal has priority since the coverage is one of the most important qualities of service that sensor networks provide. Full coverage can be guaranteed by calculating the sensing radius using Voronoi diagrams.

The second goal aims at saving energy of sensors. As in our assumptions, the algorithm is performed by a central node with unlimited energy. The energy spent on the computation of our algorithm will not be considered in our energy consumption calculations. Each mobile sensor node usually has several components, such as, a CPU with memory, a communication module, a sensing module, a mobility module and other functional modules like GPS for self-positioning. All of these components will draw their power from a single source with limited power supply. In our algorithm, we mostly focus on the energy consumed by the mobility and sensing modules. This is mainly because the algorithm will not increase the amount of calculation of the sensor nodes, and very little communication overhead is involved.

The sensors in this algorithm will only be required to move, after being deployed; therefore, the energy consumed in moving the sensors is a short-term consideration. However, compared to the long-term energy consumption used in sensing, the energy for moving the sensor is much larger. For example, a smoke alarm or a light sensor consumes 0.1 mWatts [90] which means it consumes 0.36 Joules per hour. Whereas, a small robotic platform invented by Mei, et al., [91], consumes more than 9 Joules per meter when moving in a straight line. Therefore, the energy used to move the sensor one meter is equal to the energy used for sensing for more than an entire day. If the terrain is not flat, it will cost even more energy for the nodes to move around. Therefore, both energy components must be taken into consideration.

5.2.3 Fitness function design

In order to maximize the lifetime of a sensor, the energy consumed must be minimized. The relationship between a sensor's lifetime, *L,* and the energy consumed by sensors can be described as follows:

$$E_{total} = E_M + E_S + E_{other} = E_M + L \cdot P_S + L \cdot P_{other} \tag{5.5}$$

where E_{total} is the total battery energy, E_M is the energy consumed in movement, E_S is the energy consumed in sensing and E_{other} is the energy used for other things, such as, computation and communication. The energies E_S and E_{other} can be represented by the sensor's lifetime L and the corresponding powers P_S and P_{other} as shown in (5.5).

A sensor's lifetime L can be estimated using the known battery capacity and the approximate energy usage. For example, let's assume the battery inside of a sensor node can supply a total of 10,000 Joules. Also, if this sensor moves 50 meters on average spending 8 Joules/meter, the total energy for moving the sensor will be 50*8= 400 Joules. Assume the power consumed for sensing is around 0.2 Watts and the power for other activities is 0.3 Watts, we can calculate the lifetime *L* by doing the following:

$$10000 = 50 \cdot 8 + L \cdot 0.2 + L \cdot 0.3 \tag{5.6}$$

$$L = 32000\ Seconds \tag{5.7}$$

We define an equivalent estimation of power P_M for movement, which is related to the estimated lifetime *L* and energy E_M consumed in movement:

$$E_M = P_M \cdot L \tag{5.8}$$

Therefore, equation (5.8) can be substituted in equation (5.5) and derive:

$$L = \frac{E_{total}}{P_M + P_S + P_{other}} \tag{5.9}$$

In order to maximize *L*, $P_M + P_S$ must be minimized as E_{total} and P_{other} are both constant.

According to assumption 5 in Section 5.2.1, if any sensors die because of battery exhaustion, the sensor network cannot maintain its functionality. Thus, to prolong the lifetime of the entire sensor network, we have to prolong the lifetime of every sensor. In fact, the sensor that drains its battery the fastest will create a bottleneck in prolonging the entire sensor network's lifetime. Therefore, we design the algorithm to maximize the lifetime of the sensor node that has the shortest life. This sensor has the maximum power consumption among the other sensors in the network. These objectives can be achieved by minimizing the fitness function:

$$F = Max(P_M + P_S) \tag{5.10}$$

In the redeployment process, according to the energy model for mobile sensors in Section 3.4, P_M is proportional to the sensor's traveled distance, D_{total}, that is calculated by equation (3.20). P_S is related to the sensors' sensing radius R_S, as described by equation (3.24) or (3.25) for the linear or quadratic models respectively, from Section 3.5. We calculate R_S using the Voronoi diagram to ensure the entire sensing field can be covered.

Voronoi diagrams divide the sensing field into subareas. There is one sensor in each of the subareas, which is comprised of all points closer to that sensor than all others. If each sensor can cover the subarea it belongs to, the entire sensing field will be covered. Therefore, in order to cover an entire sensing field, the sensing radius of each sensor must be set equal to the distance between the sensor and the furthest Voronoi diagram subarea vertex:

$$R_S = Max(dv_j) \tag{5.11}$$

where dv_j is the distance between a sensor node and the subarea vertex.

An example of Voronoi diagram is shown in Figure 5.9. Five sensors S_1, S_2,…, S_5 are deployed in the sensing field. Therefore, the sensing field is divided into five subareas by the Voronoi diagram. The blue straight lines indicate the edges of the subareas. Consider the subarea surround sensor S_3 in the middle. It has four vertexes. Among these four vertexes, v_4 has the longest distance from the central node S_3. Therefore, in this case, the sensing radius of S_3 should be dv_4.

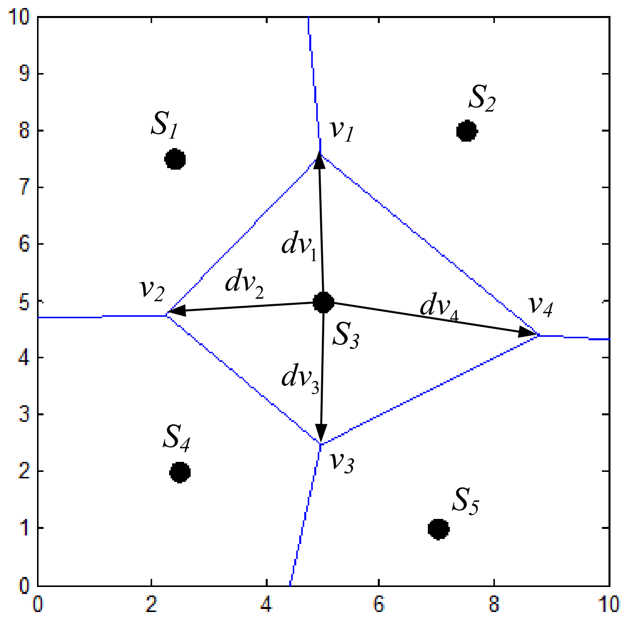


Figure 5.9 Voronoi diagram and sensing radius calculation.

According to the above, the fitness function equation (5.10) can be obtained using the travelled distance and sensing radius as follows:

$$F = Max(\frac{k_{dis}}{L} D_i + k_l R_{Si}) \tag{5.12}$$

$$F = Max(\frac{k_{dis}}{L} D_i + k_q R_{Si}^2) \tag{5.13}$$

where $i = 1,2,\ldots N$ and N is the total number of sensors, D_i and R_{Si} denote each sensor's travelled distance and sensing radius, respectively, L is the lifetime of the sensor network and k_{dis}, k_l, and k_q are energy related coefficients described in Section 3.4 and 3.5. The first fitness function in equation (5.12) is used for linear energy model of range adjustable sensors, and equation (5.13) is used for quadratic energy model.

The normalized fitness function below is derived to assist in designing the weight coefficients. At first, we rewrite the fitness function using the normalized travelled distance and sensing range:

$$F = Max(\frac{k_{dis}}{L} \cdot D_{est} \cdot \overline{D}_i + k_l \cdot R_{est} \cdot \overline{R}_{Si}) \tag{5.14}$$

$$F = Max(\frac{k_{dis}}{L} \cdot D_{est} \cdot \overline{D}_i + k_q \cdot R_{est}^2 \cdot \overline{R}_{Si}^2) \tag{5.15}$$

where $\overline{D}_i$ is normalized sensing travelled distance, $\overline{R}_{Si}$ is the normalized sensing range. They can be calculated using the estimated maximum travel distance D_{est} and estimated maximum sensing radius R_{est}:

$$\overline{D}_i = \frac{D_i}{D_{est}} \tag{5.16}$$

$$\overline{R}_{Si} = \frac{R_{Si}}{R_{est}} \tag{5.17}$$

For simplifying the fitness function and normalizing the weight, the fitness functions are rewritten as follows:

$$F = Max(w_1\overline{D}_i + w_2\overline{R}_{Si}) \tag{5.18}$$

$$F = Max(w_1\overline{D}_i + w_2\overline{R}_{Si}^2) \tag{5.19}$$

For simplifying the fitness function and normalizing the weight, the fitness functions are rewritten as follows:

$$w_1 + w_2 = 1 \tag{5.20}$$

Also, comparing equations (5.18) and (5.19) with (5.14) and (5.15), it can be found that w_1 and w_2 also have to follow a certain relationship:

$$w_1 : w_2 = \left(\frac{k_{dis}}{L} \cdot D_{est}\right) : \left(k_l \cdot R_{est}\right) \tag{5.21}$$

$$w_1 : w_2 = \left(\frac{k_{dis}}{L} \cdot D_{est} \right) : \left(k_q \cdot R_{est}^2 \right) \tag{5.22}$$

Equation (5.21) is used to normalize fitness function (5.18) for linear energy model, and (5.22) is used in (5.19), which is used for the quadratic energy model for range adjusting.

5.2.4 Particle Swarm Optimization

As introduced in Section 2.2.2, the particle swarm optimization (PSO) algorithm is an evolutionary optimization algorithm developed for optimizing fitness functions. In our algorithm, PSOs are used to find the minimized value of fitness function described in (5.18) and (5.19).

During a PSO, an initial group population of candidate solutions will be generated first. In each generation a fitness function will be evaluated for each of the candidate solutions. The optimizer will record two types of best fitness values and the candidate solution values corresponding to them. The first type of fitness is global best (gbest), which corresponds to the best fitness among the total population. The second type is personal best (pbest), which corresponds to the best fitness each specific population has ever achieved among the generations. In each generation, the candidate solutions will move toward the gbest and pbest in random velocities. The main idea can be described as follows:

$$v_i \leftarrow \omega v_i + a_p r_p (p_i - S_i) + a_g r_g (g - S_i) \tag{5.23}$$

$$S_i \leftarrow S_i + v_i \tag{5.24}$$

where v_i is the velocity of the ith population, S_i denotes the position of it, ω is the weight related to the speed in last round, a_p and a_g are the acceleration toward the personal (p_i) and global best (g) positions, and r_p and r_g are random real numbers in [0,1]. Equation (5.23) shows the calculation of velocity and equation (5.24) shows the next position is the current position plus the velocity.

In our algorithm, the objective of the PSO is to minimize the fitness function defined by (5.18) or (5.19). The population is a vector with the positions of sensors and the sensing radius of each sensor:

$$S_i = [x_1, x_2, ..., x_N, y_1, y_2, ..., y_N] \tag{5.25}$$

where x and y are the coordinates of the sensors, and N is the total number of sensors. Also, the velocity vector has the same length as the population vector.

There are several coefficients that need to be designed in this algorithm, namely, the weight ω and the acceleration rates a_p and a_g. Also, we need to design the boundary of the population and velocity. The boundary of the population we designed here is the boundary of the sensing field. The boundary of the velocity is chosen as one quarter of the sensing field length. In this way, the algorithm will converge quickly.

5.2.5 Algorithm Procedure

The algorithm contains four main steps:

1) ***Parameter estimations:*** Before deployment, the central node estimates the parameters (ω, w_1, a_p, a_g, and the boundaries) for the algorithm using the pre-known battery capacity and power consumption of sensors components. These estimations are based on the prior knowledge of the sensors and sensing field, such as the energy capacity

of each sensor, the power consumption of different parts of each sensor node, the size of the sensing field, and the total number of sensors that are deployed.

2) ***Collection of information*****:** After deployment, the central node collects the location information of all the sensors.

3) ***Algorithm calculation*****:** The central node performs the optimization algorithm and finds out the best relocation positions for sensors. The central node sends the destination position to each sensor.

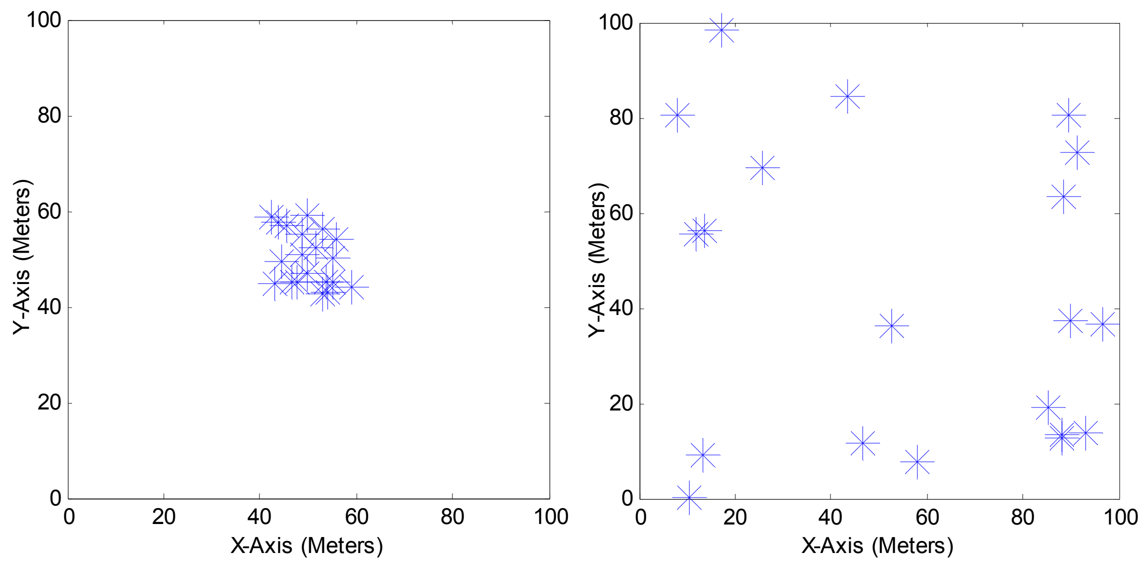
4) ***Movement to destination*****:** After receiving the relocation positions, the sensors move immediately to their destinations.

5) ***Self-healing process:*** If any sensor dies due to any reason, the coverage cannot be maintained. Therefore, a self-healing process must be undertaken, by repeating steps *1* through *4*.

5.2.6 Simulations and analysis

In our example simulation, the sensing field is 100 meter by 100 meter. Two scenarios are simulated. In both scenarios, 20 sensors are deployed. In the first scenario, sensors are initially randomly deployed near the center of the sensing field, and in the second scenario, sensors are randomly deployed throughout the entire sensing field. Initial deployments of sensors in both scenarios are shown in Figure 5.10. Both the linear and quadratic energy models for range adjustment of sensing are tested.

In order to analyze the cases in which sensing and travelling have different weight relationships, we use the range of w_1 from 0.9 to 0.1. The population size is set to 200 and the generation limit is set to 400. The two parameters a_p and a_g are both 2.



(a) Scenario 1 (b) Scenario 2

Figure 5.10 Sensor random initial deployments.

Figure 5.11 shows the outcome of our algorithm for both energy models for the two scenarios. The *x* and *y* axes respectively show the sensor's sensing radius and travelled distance that has the maximum energy consumption. Trade-offs can be seen between the sensing radius and travelled distance and they are controlled by weights w_1 and w_2. Decreasing the weight of the distance travelled (w_1) will lead to more travelled distance by the sensors and less sensing radius. When w_1 is 0.1, the sensor that has maximum energy consumption tends to have the smallest sensing radius, which is around 19 meters compared to other w_1 values. The travelled distance of that sensor is 48 and 38 meters for scenarios 1 and 2, respectively. When w_1 is 0.9, sensors do not move significantly from their initial position and the sensing radius is around 63 meters and 30 meters for scenarios 1 and 2, respectively. Figure 5.11 shows that in both scenarios, the linear and quadratic models lead to a similar optimal layout. However, it should be pointed out that the same optimal points correspond to a different weight w_1 in different energy models. Figure 5.11 also shows, in the second scenario, that sensors travelled less than in the first scenario. This is because the sensors are more uniformly distributed in the second scenario than the first scenario.

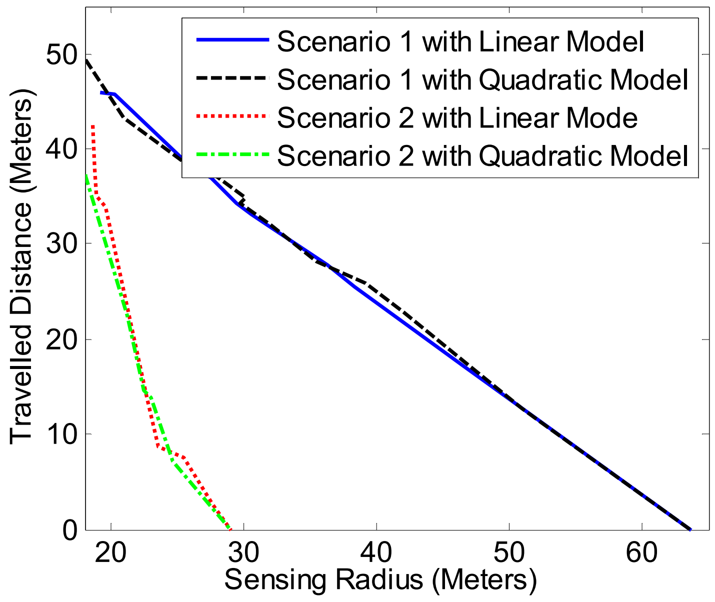


Figure 5.11 Best fitness value outcome of PSO algorithm.

Figure 5.12 is an example of the sensors' relocation after the PSO algorithm in scenario 1 has run using the linear energy model. The sensor that consumes the most energy, the sensor that has the largest sensing radius, and the sensor that travels the largest distance are shown. The sensor that travels most is the same as the sensor that consumes the most energy. It can be clearly seen that the sensor, which travels most, has smaller sensing radius than the sensors that are close to the center of the field. This is because the sensors near the center need to travel less. Also, Figure 5.12 indicates that the sensor node with the highest sensing radius is the sensor closest to the center, which needs to move the least.

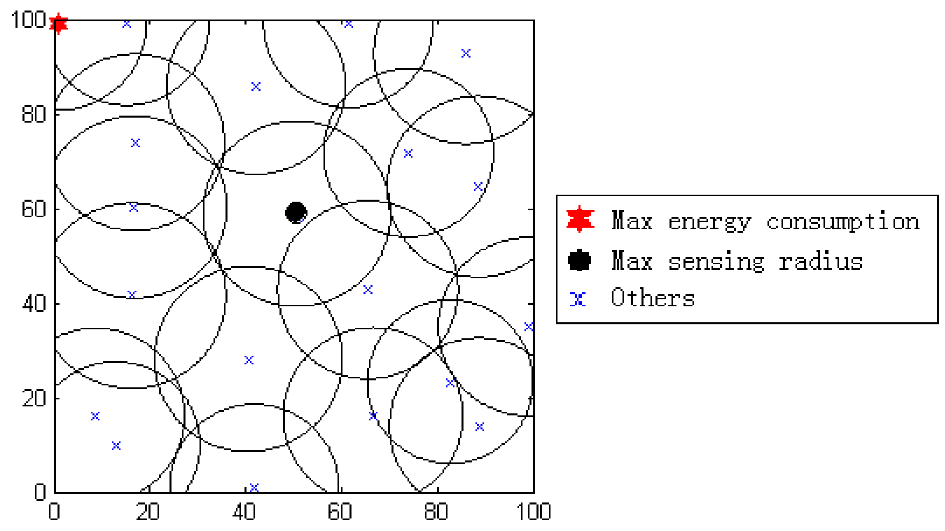


Figure 5.12 Optimal relocation simulation outcome example.

In order to see the benefit of the relocation algorithm, we calculate the energy consumption before running the algorithm and the energy consumption of the optimal outcomes after the algorithm for each case. Since the fitness function value is the representation of the energy consumption of the sensors in travel and sensing, we compare the energy savings for different weight values. The results are shown in Figure 5.13, where w_1 ranges from 0.1 to 0.9 with a 0.1 step. The savings are calculated by using difference of fitness function value divided by the fitness value of the initial deployment. For example, assume the value of the fitness function is 1.5 when sensors are initially deployed, and the best fitness function value is 0.9 after the algorithm, the savings is calculated as (1.5-0.9)/1.5 = 0.4 = 40%. Figure 5.13 shows that our algorithm can provide up to 89%, 65%, 47% and 28% energy savings in different scenarios.

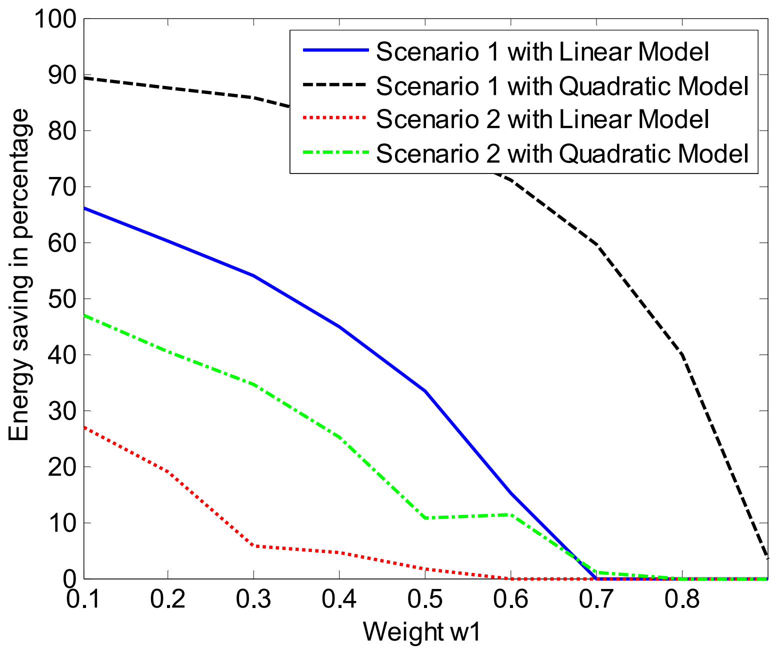


Figure 5.13 Energy savings of optimization algorithm.

Figure 5.13 shows that PSO can provide more energy savings for the lower values of the weight w_1 of distance travelled. The reason for this is that sensors move more when w_1 is smaller, and the deployment outcome changes more from the initial deployment. It also shows, in the same scenario, that the algorithm provides more energy savings for the quadratic model. This is because in the quadratic model, the sensing radius has heavier weight compared to the linear model. Therefore, the fitness function value is more sensitive to changes of the sensing radius.

As in real applications, it is not likely to have static sensors just deployed in the center part like scenario 1. However, scenario 2 is always the case when sensors are initially randomly deployed. In scenario 2, the maximum savings for the linear and quadratic models are approximately 28% and 47%, respectively. This happens when w_1 is relatively small (0.1), which means energy consumption for sensing has much larger weight than energy consumption for relocating. When w_1 is in the range 0.6 to 0.8, travelling costs significantly more energy than for sensing. Thus, sensors prefer not moving and adjust their sensing radius to prolong their lifetime.

5.2.7 Conclusions

This section introduced a centralized relocation algorithm for wireless sensor networks, which uses the Particle Swarm Optimization (PSO) aiming to prolong their lifetimes. The maximum total energy consumed is optimized. Our simulations use different weights between sensing radius and travelling energy consumption that lead to different relocation optimal layouts. Our simulation results clearly show that sensor that needs to travel more have a smaller sensing radii. Also, our results show that energy savings are more significant when the sensing energy has a heavier weight.

5.3 Hybrid PSO with distributed algorithm

As we have indicated in Section 5.2, PSO is a centralized optimization algorithm that we have used it for the purpose of minimizing the energy consumption of sensors. It can also be hybrid with the distributed algorithm for prolonging the lifetime of wireless sensor networks and the hybrid PSO will provide a better solution.

In this section, the distributed relocation algorithm for WSN we derived in Section 4.2 is hybridized with PSO is proposed. The convergence of the PSO is faster and the result will be closer to the global optimal value. This algorithm addresses the problem of maintaining full coverage for a WSN while prolonging the sensor network's lifetime by optimizing the energy consumption. First, a hybrid algorithm, which solves both the energy consumption and coverage maximization problems, is developed. Then, our distributed relocation algorithm is hybridized with the PSO algorithm in order to significantly improve the algorithm's optimization performance.

5.3.1 Algorithm Description

Our objective of this algorithm is to guarantee the entire sensing range will be covered by the wireless sensor network and also prolong the life time of the sensor network.

a) Assumptions for Hybrid PSO

Here are our assumptions and used models for the algorithm.

Full coverage is part of our optimization goal, and thus only the binary model is applied. To cover the entire sensing area with a more efficient energy usage than the fixed range sensors, the range-adjustable sensors are used here also. The sensing radius of each sensors are calculated in the same way using equation (5.11) in Section 5.2.3. The performance on both the linear and quadratic energy model for range-adjustable sensors are examined in our algorithm.

b) Fitness function design

There are two goals in our optimization:

Ensuring that the entire sensing field is covered by the sensor network. This can be achieved by applying equation (5.11).

Minimize the energy consumption of the sensors so that the lifetime of the sensor network can be maximized. This can be done by minimizing the energy consumption function:

$$E_{total} = E_M + E_S + E_{other} \tag{5.26}$$

where E_{total} is the total energy stored in each sensor's battery and E_M, E_S and E_{other} are the energy consumed by the sensor's relocation, sensing and other activities such as communication, computation, self-localizations.

The energy can be expressed by the lifetime L and power:

$$E_{total} = E_M + L \cdot P_S + L \cdot P_{other} \tag{5.27}$$

where P_S is the sensing power which can be calculated by equation (3.24) or (3.25) depending on the energy model and P_{other} can be estimated by the type of sensor used. Sensor lifetime L can be rewritten as:

$$L = \frac{E_{total} - E_M}{P_S + P_{other}} \tag{5.28}$$

In a wireless sensor network, if any sensor dies, a relocation optimization will be required. Therefore, we define the fitness function as the lifetime of the sensor that has the shortest life:

$$F = Min(L) \tag{5.29}$$

$$F = Min\left(\frac{E_{total} - E_M}{P_S + P_{other}} \right) \tag{5.30}$$

c) Hybrid PSO

The detail of traditional PSO has been described in Section 5.2.4. The main process can be described in the following process:

$$v_i \leftarrow \omega v_i + a_p r_p (p_i - S_i) + a_g r_g (g - S_i) \tag{5.31}$$

$$S_i \leftarrow S_i + v_i \tag{5.32}$$

Here, our relocation algorithm presented in Section 4.2 is hybridized with PSO to provide enhanced optimization results. This is based on the fact that distributed relocation algorithms naturally increase the coverage and energy savings of sensors because they redistribute the sensors more evenly.

The relocation algorithm uses relative locations of sensors in order to relocate the sensors more uniformly in the sensing field. This algorithm in our PSO will help decrease the sensing radius of the sensors.

In the distributed relocation algorithm, each sensor divides its neighboring area into six subareas and finds out the nearest neighboring sensor in each subarea. It creates two vectors pointing from itself to the closet and farthest sensor from those six sensors:

$$\overrightarrow{d_s} = A_s \angle \varphi_s \tag{5.33}$$

$$\overrightarrow{d_l} = A_l \angle \varphi_l \tag{5.34}$$

where $\overrightarrow{d_s}$ and $\overrightarrow{d_l}$ are the vectors to the closest and farthest sensor respectively. A and φ represent the amplitude and angle of the vectors. Virtual image nodes will be used when a sensor is close to the sensing field boundary.

Then, a moving vector is defined for each sensor as below:

$$\overrightarrow{m} = (A_s - d_o)\angle \varphi_s + (A_l - d_o)\angle \varphi_l \tag{5.35}$$

where d_0 is a threshold distance between sensors and we set it to be $\sqrt{3}\,R_S$ here. We also define a moving vector of the entire group of sensors:

$$\overrightarrow{M} = [\overrightarrow{m_1}, \overrightarrow{m_2}, \ldots, \overrightarrow{m_N}] \tag{5.36}$$

It can be seen that both vectors v_i and $\overrightarrow{M}$ have the same physical meaning in terms of sensor relocation. Specifically, v_i is related to global and individual optimization positions obtained by any PSO and $\overrightarrow{M}$ arises from the natural properties of redeploying the sensor network. Therefore, these vectors are combined in order to improve the performance of traditional PSO. This hybridization is formulated by the following equation:

$$v_i \leftarrow \omega v_i + a_p r_p (p_i - S_i) + a_g r_g (g - S_i) + a_M r_M \overrightarrow{M} \tag{5.37}$$

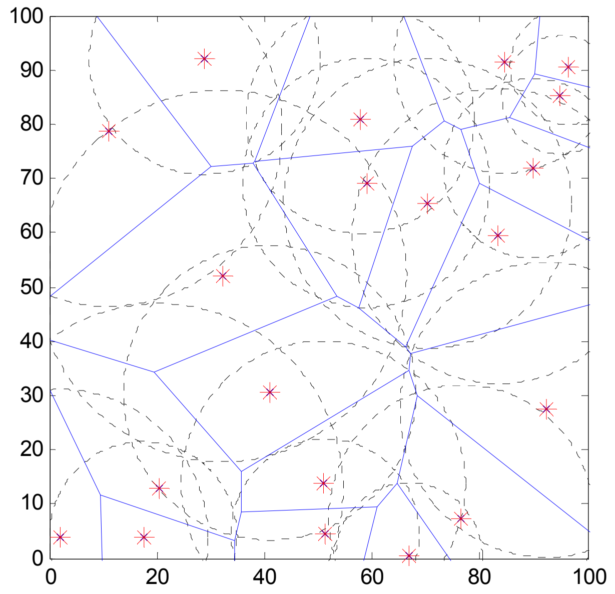
This will be used in our hybrid PSO instead of (5.31). In our algorithm, ω is 0.5, both a_p and a_g are 2, and a_M is 1.

5.3.2 Simulation and Results analysis

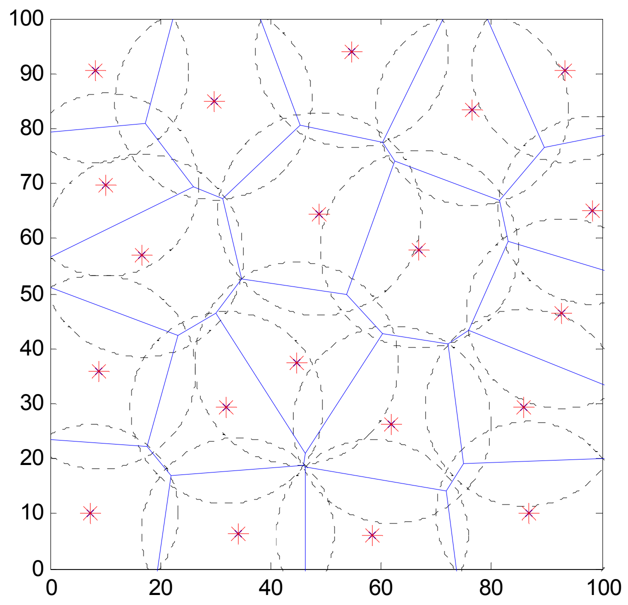
In our simulation, sensors are deployed in a 100 meters by 100 meters field. The initial energy related coefficients are the same. Assume that two rechargeable AA batteries contain 1,600 mAh energy are used in each sensor, so the total energy for each sensor will be 17,280 Joules. Sensors use the low energy cost mobile device [9] to carry the sensors, so k_M is 8 J/m. For a typical sensor with 0.2 watts in a range of 10 meters, k_l will be 0.02 W/m when the linear model is used, and k_q will be 0.002 W/m^2 when the quadratic model is used. Other energy consumed by the CPU and other components is 0.5 watts.

Three initial deployments are examined which 20, 30, and 40 sensors initially deployed. All sensors are randomly deployed in the sensing field. Both linear and quadratic models are applied for each case. In order to evaluate the performance of the hybrid PSO, we compare the results with the traditional PSO and Genetic Algorithm (GA) which is another centralized heuristic search comparable with PSO. The same initial populations are taken. The population sizes of all cases are 50 and the maximum generation number is set to be 100. As it is indicated in Section 5.2.6, when we analyzed the performance of traditional PSO, we use 200 populations and 400 generations. The different settings here also show that the hybrid PSO has significant improvement beyond the traditional PSO.

Figure 5.14 shows an example of redeployment outcome when hybrid PSO is used. In this example, the linear energy model is applied. Figure 5.14 (a) shows the initial deployment of 20 sensors. All sensors are randomly deployed. It can be seen that with random deployment, some sensors have to use large sensing ranges, two to four times larger than the range of other sensors. This situation causes the lifetime of the sensor network to be short, since some of the sensors drain their batteries much faster than other ones. In Figure 5.14 (b) we can see that after optimization, sensors are more evenly distributed. Therefore the lifetime of sensor network can be prolonged.



(a) Initial deployment



(b) Optimization outcome

Figure 5.14 Optimization example of 20 sensors using linear energy model

Simulation results are shown in Figure 5.15 and Figure 5.16 for linear and quadratic models separately. The convergence process is shown. A significant increase of the lifetime of the sensor networks can be seen in both energy models after optimization. It can be seen that both the PSO and the hybrid PSO have the ability to optimize the energy consumption; therefore, they provide a longer lifetime to the sensor network. Both algorithms converge to the optimal result in less than 40 generations. However, the hybrid PSO achieves 4%~20% longer higher optimal lifetimes than the traditional PSO algorithm.

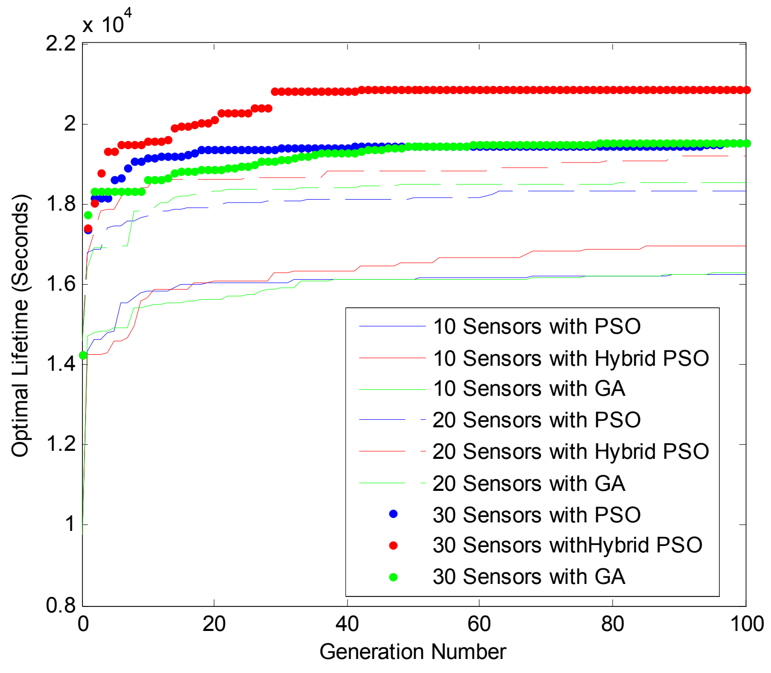


Figure 5.15 Lifetime optimization when the linear energy model is applied.

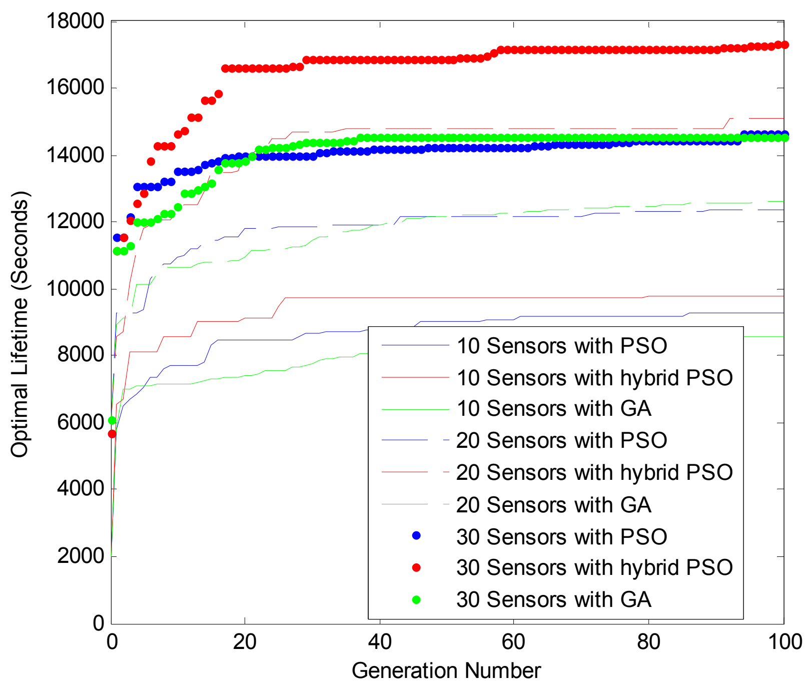


Figure 5.16 Lifetime optimization when the quadratic energy model is applied.

The distributed relocation algorithm, which has been hybridized to a PSO, moves the sensors to more evenly distributed locations, thereby requiring sensors with the shortest sensing range. This in turn yields energy savings and prolongs the lifetime of sensor networks. Therefore, the hybrid PSO will provide better improvement when energy spent on sensing is larger than energy spent on movement. This corresponds to sensor networks that have long lifetime. For example, when the linear model is used, it can be seen that when there are 30 sensors (case with the longest lifetime), the hybrid PSO has 7% longer lifetime than the one of traditional PSO. This increase in lifetime is larger than the corresponding lifetime increase for the other two cases with 10 and 20 sensors.

The lifetime extensions achieved by both algorithms for both models of optimization are listed in Table 5.1. It can be seen that the optimization process can significantly increase the lifetime of the sensor network, especially in instances when the initial deployment results in short remaining lifetime (e.g., 385% increased lifetime was achieved when the original lifetime was only 2020 seconds). Also, when the required sensing range is decreased, the quadratic model exhibits more energy savings and longer lifetime than the linear model, due to its quadratic dependence on the sensing range. This optimization algorithm can also be used throughout the entire lifetime of the sensor network. Therefore, the location of sensors will be optimized according to the remaining energy of each sensor. When any sensor dies, the algorithm can be performed so that the remaining sensors can still provide maximum possible service as long as possible.

Table 5.1 Lifetime optimization result.

Number of sensors	Linear Energy Model			
	Original Lifetime	Increased percentage from Original lifetime		Increased percentage by Hybrid PSO compare to PSO
		PSO	Hybrid PSO	
10	9767 s	66%	74%	4.3%
20	14588 s	26%	32%	5.1%
30	14247 s	37%	46%	7.0%
	Quadratic Energy Model			
	Original Lifetime	Increased percentage from Original lifetime		Increased percentage by Hybrid PSO compare to PSO
		PSO	Hybrid PSO	
10	2020 s	361%	385%	5.3%
20	6079 s	103%	148%	22%
30	5683 s	163%	190%	10%

5.3.3 Conclusions

This paper introduced a hybrid PSO optimization algorithm for prolonging the lifetime of mobile wireless sensor networks. Simulation results show after relocation optimization by hybrid PSO, sensors are more evenly distributed therefore providing a longer lifetime of the entire sensor network. Compared to traditional PSOs and GAs, the hybrid PSO can provide 5% to 20% more lifetime. The lifetime of the sensor network can be extend more than 30% when linear energy model is used and more than 150% when the quadratic energy model is used. The lifetime of the sensor network is highly depend on the initial deployment.

5.4 Summary

This chapter introduced three centralized optimization algorithms aiming at saving the energy of wireless sensor networks and prolonging their lifetime while providing optimal coverage. All of the algorithms require large amount of computation, therefore they can only be applied in the powerful central node, and not in each sensor node.

These three optimization algorithms focus on applications with two types of hardware. The Genetic Algorithm based optimization is applied to sensors with fixed sensing range and provides the optimal tradeoff between sensing coverage and the distance travelled of sensors. The Particle Swarm Optimization and hybrid Particle Swarm Optimization algorithm uses range adjustable sensors to ensure optimal coverage of the sensing field and optimal tradeoff between sensing range and travelled distance of sensors so that the lifetime of sensor network can be maximized.

The Genetic Algorithm can be used to estimate the best tradeoff before the sensor relocation occurs. The application can choose the target percentage of coverage and use the optimum outcome to decide the relocation destinations.

Both Particle Swarm Optimization and Hybrid Particle Swarm optimization focus on extending the lifetime of wireless sensor networks by making savings in terms of both long term energy and short term energy consumption. Both algorithms can extend the lifetime of wireless sensor networks by more than 60%. Our hybrid Particle Swarm Optimization algorithm can achieve more than a 5% lifetime extension compared to the traditional Particle Swarm Optmization.

Furthermore, all algorithms can continue to optimize the sensor network lifetime, in the event that sensors die due to the exhaustion of their battery power. This means that both optimization algorithms are self-healing and robust.

CHAPTER 6

DISTRIBUTED SENSING RANGE ADJUSTMENT

All the algorithms discussed above only consider equal, non-adjustable range sensors. This constraint makes for algorithms that cannot reach a goal of simultaneously providing 100% coverage and energy-efficiency, because there is too much overlap in the sensor ranges. In our algorithm, we developed a distributed method of using range-adjustable sensors for providing 100% coverage, as well as saving the energy for prolonged sensing. Our algorithm can be used either in a multitude of different cases, including directly after random deployment, after selecting and grouping sensor nodes or after the relocation of sensors.

6.1 Assumptions

In addition to the assumptions we declared in Section 3.7, we have two more assumptions to our distributed sensing range adjustment algorithm:

1. All sensors have adjustable sensing radius. Each sensor can shorten their sensing radius by lowing the power they use.
2. All sensors know their current locations within the sensing field, and the details of the sensing field, such as the coverage area boundaries, are pre-installed into their memories

6.2 Voronoi Diagram

The Voronoi diagram is a method for partitioning an area. If a set of *N* points is in the area *A*, the area will be divided into *N* subareas that each have only one point inside. Any location $L(x_l, y_l)$ in the *i*th subarea (A_i), will have the shorter distance to the point $I(x_i,$

y_i) in that subarea than compared to any other point $J(x_j, y_j)$. This can be formulated using the following equation:

$$A_i = \left\{ l \in A \mid D_i \leq D_j, \textit{for all } i \neq j \right\} \tag{6.1}$$

where D_i is distance between L and I and D_j is the distance between L and J.

In our work, the area A will be our target sensing field required to be covered, and the points will be our sensors providing the coverage. From assumptions 3 and 4 above, each sensor can compute the Voronoi diagram locally by broadcasting their location and receiving the location information from other sensors. Therefore Voronoi diagrams can be used to decide the partitions of the area closest to each sensor. Voronoi diagrams have been used in mobile sensor networks for relocation in [12]. Two sensors that share the same Voronoi subarea vertex are called neighbors.

6.3 Algorithm description

In previews algorithms that use equal and non-adjustable range sensors coverage holes and redundant coverage always exists (see Figure 6.1). Six equal-range sensors are deployed to cover a rectangular area. The area in green is covered by three sensors, i.e., the coverage is redundant. The area in red is not covered by any sensor and represents a coverage hole.

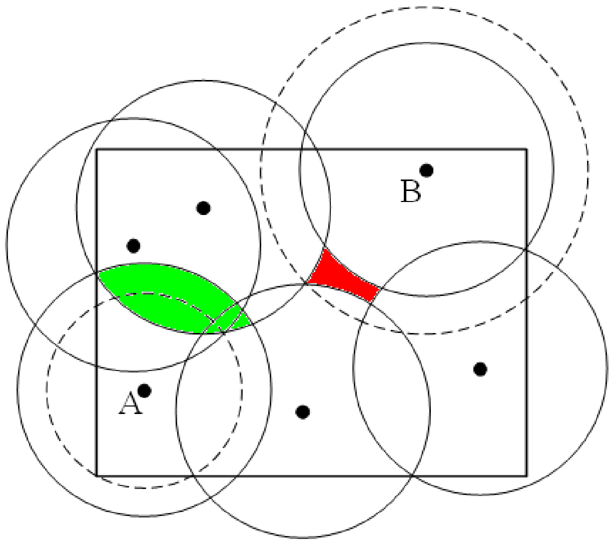


Figure 6.1 Redundancy and Coverage holes.

Our algorithm has addressed the following question: how can we use range adjustable sensors to avoid coverage holes and redundant coverage? Our first goal is to maintain full coverage, since this is required by many surveillance applications. Our second goal is to shorten the sensing range, such that, redundancies would be reduced thereby saving energy. For example, the problem shown in Figure 6.1 can be solved if the coverage can be adjusted to the dashed lines shown for sensors A and B. In this way, sensor A saves some energy, but will not create any uncovered areas. Additionally, sensor B uses more power to ensure that the sensing area is fully covered.

6.3.1 Divide and Conquer

In distributed algorithms, it is not easy for each sensor to know if the entire field is covered. A more direct and easier way to monitor the coverage ratio of the entire field is to partition the sensing field and have every sensor be responsible for covering only the area close to it. This method of "divide and conquer" is implemented using Voronoi diagram as described above.

Theorem 1: A sensing field is completely covered if, and only if, all of the Voronoi subareas that make it up are completely covered.

This can be seen in the fact that the entire sensing field is covered completely by the Voronoi diagram subareas. Additionally, from the definition of Voronoi diagram, each subarea is completely exclusive of all others. Thus, any area belonging to a Voronoi subarea not covered will result in an uncovered area of the sensing field. If all subareas of the diagram are covered, then the entire sensing field has 100% coverage.

Theorem 2: A sensor can cover its entire Voronoi subarea if it can cover all the vertexes of the subarea it belongs to.

This can be achieved by setting the sensing radius as the distance to the farthest vertex from the sensor. Sensor *I*, at location (x_i, y_i), will set its sensing radius R_{Si} according to the following equation:

$$R_{Si} = \max\left(\sqrt{(x_i - x_{vj})^2 + (y_i - y_{vj})^2}\right),\ j = 1, \ldots, K \tag{6.2}$$

where (x_{vj}, y_{vj}) are the locations of the vertices of the subarea it belongs to, and K is the total number of vertices of that subarea. Also, we consider the boundaries of the sensing field as edges of Voronoi subareas, so that each area is a closed area.

6.3.2 Sensing range shortening

In the previous section, we described a method for accomplishing our first goal of achieving complete coverage. However, solely following that method leads to wasted energy. An example of this is shown in Figure 6.2. Five sensors are deployed in a 10 by 10 square meter sensing field. The red lines indicate the farthest vertex for each sensor. Sensors adjust their sensing range according to equation (6.2). Although the entire sensing field is covered by the five sensors, Voronoi subarea for sensor *D* is mostly covered by other sensors surrounding it. Sensor *D* can just turn down its sensing range to the distance $\overline{\mathrm{DM}}$, in which *M* is the intersection point of the sensing ranges of sensors *B* and *E*. On the contrary, if any of sensors *A*, *B*, *C* and *E* use a shorter sensing radius, some parts of the sensing field will be uncovered. We define certain sensors as important to distinguish them from others and treat them differently within our algorithm.

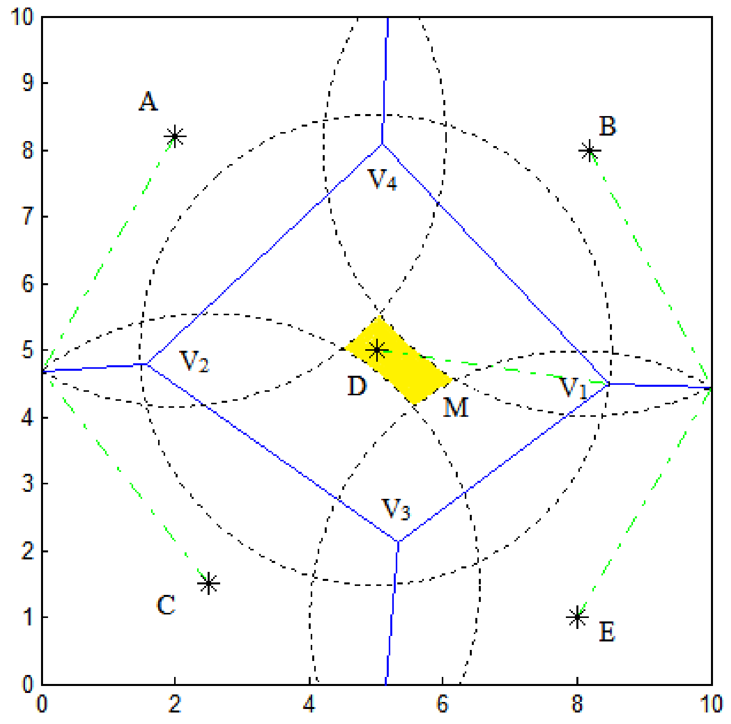


Figure 6.2 Voronoi diagram based sensing radius choosing.

Definition 1: A sensor is important if reducing any portion of its sensing range leads to some uncovered area in the sensing field.

In our algorithm, we consider a sensor as "important" if it can be described by any of the following three cases which are shown in Figure 6.3:

1. If a sensor's sensing radius is equal to the distance between itself and a corner of the sensing field, this sensor is "important" (Sensor A in Figure 6.3)
2. If two neighboring sensors set their sensing range to cover a shared subarea vertex on the sensing field boundary, then both sensors are considered "important" (Sensors B and C in Figure 6.3)
3. If three neighboring sensors set their sensing range to cover the same point, all three sensors are considered "important". (Sensor D, E and F in Figure 6.3)

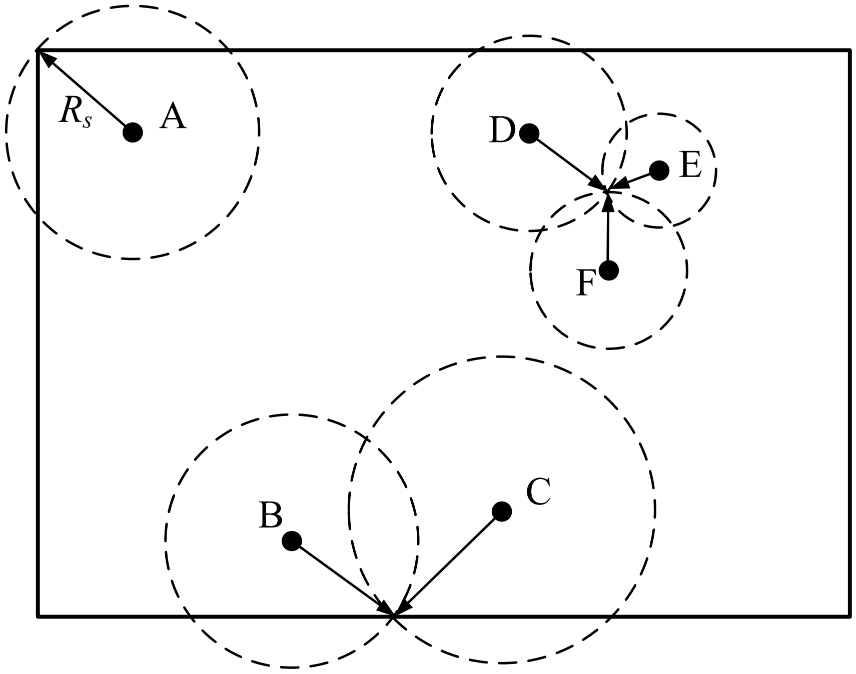


Figure 6.3 Sensor important definition.

All other sensors that do not meet these conditions are considered "not important". Our algorithm seeks to iteratively convert all "not important" sensors into important sensors by shortening sensing radius to the minimum possible thereby providing an energy savings.

In order to avoid the problem of two sensors simultaneously shortening their sensing radius, which could subsequently result in the creation of uncovered areas, each sensor performs the range shortening process individually one at-a-time based on a set of priorities.

Definition 2: The priority of the *i*th sensor is defined as the following:

$$pr_i = \frac{1}{L_i} \tag{6.3}$$

where L_i is the estimated lifetime of the sensor which is related to its remaining battery energy and sensing radius. A sensor with larger sensing radius will have shorter lifetime, and therefore, larger priority (according all sensors have the same battery and components).

This definition of the priority is to have all sensors consuming energy more evenly. A sensor closer to dying will have a higher priority to shorten its radius.

6.3.3 Detailed range-adjustment process

The entire sensing range adjustment follows the steps below.

1. Step 1: Information collection and sensing range initialization
   a) Each sensor broadcasts its locations and uses the location information it obtains to create a local Voronoi diagram around itself.
   b) Each sensor adjusts its sensing radius to cover its own Voronoi diagram using equation (6.2)
   c) Each sensor broadcasts the initial sensing radius it receives.
   d) Each sensor calculates if it is “important” and broadcasts its decision.
2. Step 2: Redundant sensor filtering

a) Each sensor checks if its entire covered area is covered by any other sensor by the information they get from step 1.

b) If yes, it turns itself to sleep and notifies all the other sensors of the change in its status.

c) Other sensors that are still on working mode will redo the Voronoi diagram calculation and range adjustment initialization.

d) This step is repeated until no sensor is redundant.

(In the simulation, we found that step 2 is not necessary when the sensors are evenly distributed.)

3. Step 3: Priority setup and making a waiting list

Here, sensors will decide if they will start the sensing range shortening process or if they will wait for other sensors. Each “not important” sensor creates a waiting list including each of its neighboring sensors with a higher priority that have not completed their sensing range adjustment. Only if its waiting list is empty can a sensor start the process of shortening its sensing radius, with the updated sensing radius of its neighboring sensors.

4. Step 4: Range shortening

Once a sensor finds out that there is no other sensor on its waiting list, it starts the sensing range shortening process.

a) A candidate solution list is created by each sensor. The candidate solutions are initialized as the distances from the sensor to all the subarea vertices. For ease of explanation, we name these vertices as V_1, V_2, V_3…, from farthest to closest from the sensor. The corresponding values of distance are v_1, v_2, v_3…. Our initial solution is v_1, but in the case that the sensor is not important, we need the solution value until the sensor is considered important. This is depicted for sensor D in Figure 6.2.

b) The sensor finds out which subarea vertex is farthest from it, which is V_1 as shown in Figure 6.2.

c) Since each vertex is surrounded by exactly three sensor nodes, it then finds the other two neighboring sensors in the subareas that share vertex V_1, which are sensors B and E in Figure 6.2.

d) The sensor first checks if its location is within the sensing range of both neighboring sensors (B and E). If yes, v_1 will be changed to 0.

e) If 4d is not met (if the sensor is outside of the neighboring sensors' range), then the sensor calculates if the circles of the two neighboring nodes' sensing ranges intersect. If not, the sensor calculates what the intersection points would be if it selected a larger sensing range among relevant points on the edges of the subarea. There should be at least one intersection point because the sensor is not redundant, as we determined in step 2. Then it calculates the distance between the sensor and the intersection points that are on the edges next to V_1(if any) and selects the larger value to replace the value of v_1. If neither of the intersection points is on either of the two edges, we conclude that V_1 is completely covered and set v_1 to be 0.

f) If the sensing ranges of two neighboring nodes intersect, calculate the distance from the sensor to the closer neighboring node and replace the value of v_1 with this distance, only if it is less than the original v_1.

g) In parts 4a through 4f, we modify one value in the candidates of solution values. We now set our solution as the largest value among the candidate solutions. Check if the sensor is now important. If yes, go to step 5, otherwise, repeat part 4b through 4g for other vertices from largest to smallest. For example, it can be seen that if part 4f is met and that value is larger than v_2, the sensor is important after the sensing radius has been set to v_1. However, in the case of Figure 6.2, all values of v_1 to v_4 will be modified to the circle intersections, and at last we choose the distance $\overline{DM}$. This is because eventual values of v_2, v_3, and v_4 are all smaller than that of v_1. As such, the repetitions of steps 4a through 4g for each candidate is needed to modify the solutions until the largest modified value, in this case v_1, is picked as final solution in an attempt to make the sensor important.

h) If after all candidate solution values, have been modified, and the sensor is still not important, then set the sensing radius to be 0 and go on to step 5.

5. Step 5:Finish

After finishing adjusting the sensing radius, each sensor will broadcast a message of its new radius. Therefore, the other sensors that were waiting on that sensor can delete the sensor from their waiting list and update the new radius for use in their adjustment process.

In the above steps, we focus mainly on the method of reducing the sensing range while maintaining the coverage for the sensors in the interior of the sensing field. In the case that a sensor node is close to the boundaries of the sensing field, we treat those boundaries as Voronoi subarea edges made by the sensor node and some virtual nodes outside the sensing range with zero sensing radius. In this way, we can perform the steps listed above to all sensors.

6.4 Simulation and Results

In order to evaluate the performance of our algorithm, simulations were conducted with two different scenarios. In both of the scenarios, sensors are initially, randomly deployed. In the first scenario, we perform a potential field algorithm as discussed in [7] first, and then apply our sensing range adjustment. In the second scenario, we directly perform our algorithm. Since the initial deployment is random, we perform the algorithm significantly large amount of times and then analyze the average outcomes. In each scenario, we developed three cases with 20, 30, and 40 sensors deployed respectively. We consider situations of both continuous and discrete range adjustment. The discrete ranges are in the set [1, 2, 3,…] meters.

6.4.1 Sensing radius adjustment

Figure 6.4 shows the results of both scenarios for continuous range adjustments. As our first goal is to reach 100% coverage, we compare our results of average sensing radius after our algorithm (black lines) with the value that originally was necessary in order to get 100% coverage (blue lines). For example, in scenario 1, if all sensors are with equal sensing radius, in order to cover the entire sensing field, a 20.9 meters sensing radius on average is required after 10000 runs of the simulation. After we perform our range adjustment process, we can cover the area using an 18.2 meters sensing radius. It can be seen, in both scenarios, that our algorithm can greatly reduce the sensing radius, compared to when the equal sensing radius sensors are used. On average, the sensing radius can be shortened by 13% - 20% for scenario 1, and 55% to 58% for scenario 2. The improvement is very significant in scenario 2, when sensors are randomly deployed without any prior relocation schemes. This is because the density of the sensors cannot be balanced under these circumstances, and a very large sensing radius is required to cover the entire sensing field when initially using the same sensing range for all sensors.

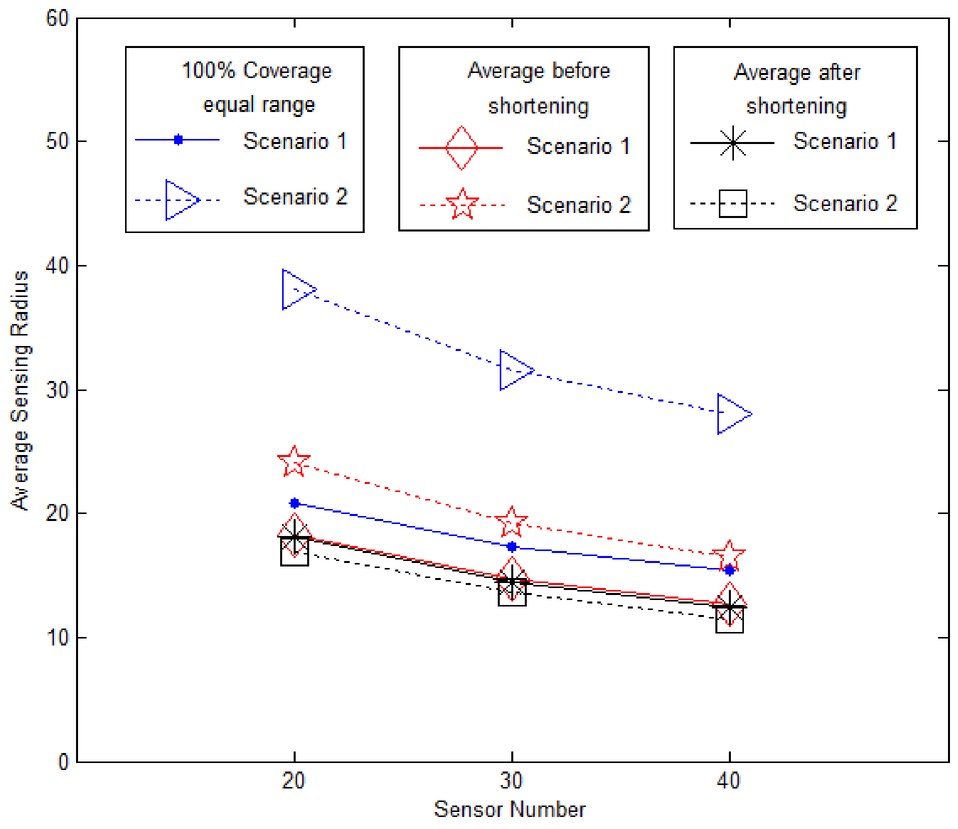
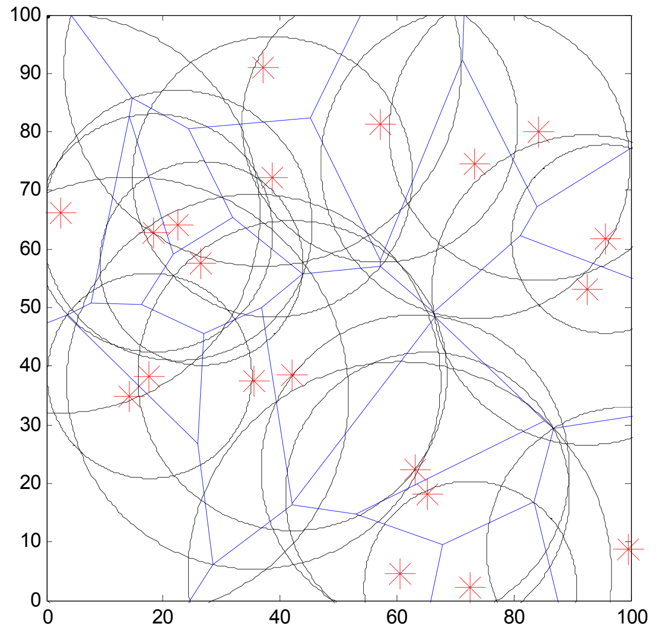
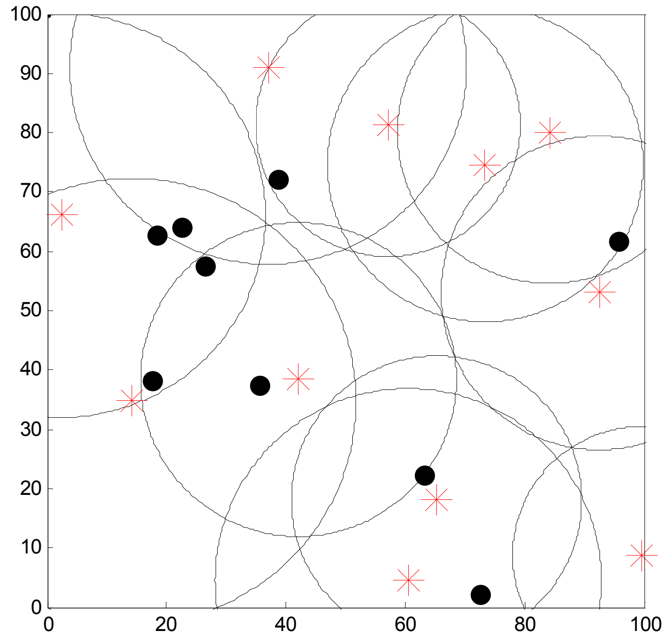


Figure 6.4 Simulation outcomes with different number of sensors

The simulation results for discrete range sensors are very similar to the ones we get from the continuous range adjustable sensor. In the discrete range case, the sensors need to use 0.4 meters more on average compared to the continues range. In theory, if the sensing range is uniformly distributed, the difference between the continues value and the rounded-up discrete value is 0.5 meters on average. Our results obtain 0.1 meters less because when some of the sensors round up their sensing radius, other neighboring sensors are able to use less sensing radius.



(a)



(b)

Figure 6.5 Example outcome for scenario 2.

In order to specifically evaluate our shortening process, we also show the average sensing radius after step 1 which is obtained only by Voronoi diagram before the shortening process. It can be seen that even when the sensors are distributed fairly evenly, our shortening process can still result in a shorter sensing radius on average to save some energy. With the scenario 2, the shortening process has a more significant benefit. It can provide 29%-30% savings in sensing radius on average for the entire sensor network.

6.4.2 Communication overhead

In WSNs the communication overhead is also an important part related to the energy consumption. Therefore, we examine the communication overhead of the algorithm.

From our simulations above, we found out that in the scenario 1 which applies our algorithm after using potential field, no redundant sensor are found in step 2: *redundant sensor filtering*. Therefore, we conclude that there is no redundant sensor if the sensors are evenly distributed such as after the potential field algorithm. Thus it is not necessary to do the redundant sensor filtering step in this scenario. By skipping the step 2, one less package will be send from each sensor.

In order to check the communication overhead, we calculate the average number of packets that are sent in order to run the algorithm. Figure 6.6 shows the relationship between the number of sensors and the number of packets. It can be seen that the number of packets is approximately linear to the scale of the sensor network. The packets sent in scenario 1 that includes the potential field algorithm are less than in the scenario executing our algorithm directly after random deployment. This is because in the scenario including a potential field algorithm, sensors do not need to check if they are redundant, and one less packet will be sent from each sensor.

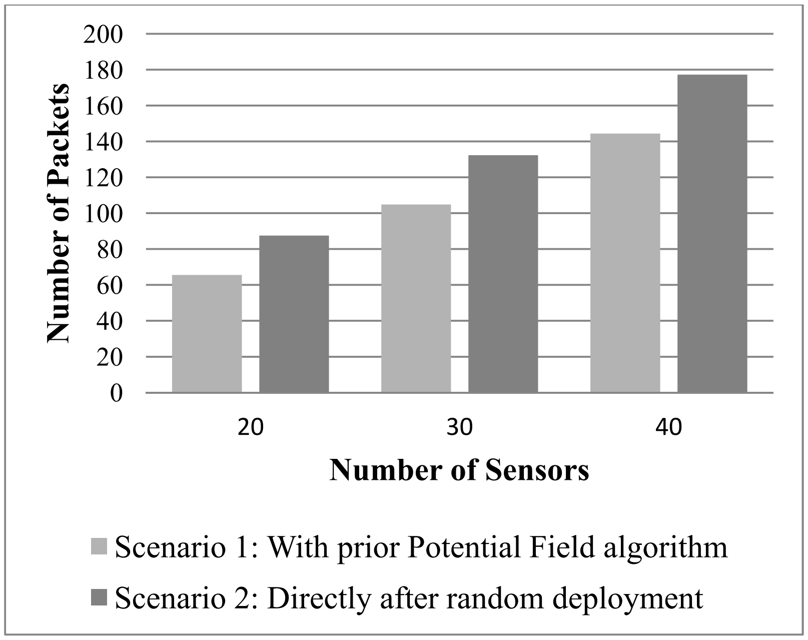


Figure 6.6 Communication overhead of algorithm

6.5 Conclusions

This chapter addressed the complete area coverage problem of WSNs. Range-adjustable sensors are used so that the coverage holes and redundant problem in non-adjustable equal range sensors networks can be solved. Up to more than 50% energy can be saved compared to the equal non-adjustable sensor networks while the coverage of WSNs can be achieved at 100%. The algorithm can work for both uniform and nonuniformly distributed sensor networks. It has more significant sensing range adjustment performance when sensors are not uniformly distributed. Also, the algorithm can be incorporate with the existing solutions with sensor relocation, scheduling or other coverage related algorithms so as to provide better coverage and energy savings.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

As stated in the beginning of the dissertation, coverage and energy consumption are two major aspects of wireless sensor networks that are considered in delivering quality of service. This dissertation addressed the coverage and the energy consumption optimization problems of mobile, wireless sensor network with different application requirements. In this dissertation, we first summarized the existing solutions for optimizing coverage and lifetime of sensor networks. Second, we described the models and calculations related to sensing coverage and energy models. Third, seven novel optimization algorithms with self-healing properties were developed. Different algorithms deal with different application requirements and hardware availabilities. Simulations for testing the algorithms have been carried out for a variety of cases.

Three different centralized optimization algorithms have been derived for sensor networks that have a powerful central node. The optimizations are based on evolutionary optimization algorithms, such as, the Genetic Algorithms (GA) and the Particle Swarm Optimization (PSO). Both are suitable for finding for solutions to non-deterministic, polynomial-time hard (NP hard) optimization problems. Compared to GAs, PSOs are easier to design and implement. With the help of the central node, the substantial amount of calculation in optimization algorithms can be executed. Simulation results have shown that optimal tradeoffs between sensor network coverage ratio and the travelled distance of the sensors with fixed sensing radius can be achieved. In other situations 100% coverage is achieved using range adjustable sensors by finding optimal tradeoffs between the sensing radius and travelled distance. This also prolongs the lifetime of the sensors. Also, significant energy savings can be achieved by enlarging the sensing radius.

Three distributed algorithms have been developed for wireless sensor networks without any central nodes. The sensors use limited information to relocate themselves cooperatively toward the ideal coverage configurations. Significant improvement of sensing coverage has been achieved in all algorithms after approximately 10 to 20 rounds of the algorithms. Our algorithms are 50% to 75% faster than the existing potential field solutions.

Also, our work considered cases in which self-localization systems are available. When sensor localization information is available, the presented algorithms have similar coverage outcomes, while typically having about 20% energy savings when compared potential field algorithms. If the sensors do not have self-localization systems or their localization systems are not accurate enough, our algorithm can still provide a significant increase in coverage ratio. This is very important because a lot of the sensing and monitoring applications happen for in indoor environments. The accuracy of these systems within these environments is around 2 meters, which is much larger than the thickness of the walls. This renders the use of location information non-applicable for indoor application optimizations because the algorithm cannot use such erroneous data provided by the sensors. However, this dissertation solves this problem. The tradeoff of not knowing the locations of sensors is the increase in travel distance. In general, this costs 25% to 30% more energy for the sensors to move.

One distributed algorithm is designed specially for range-adjustable sensors to maintain the 100% coverage and also using short sensing radius so as to save energy. This algorithm is combinable with the relocation algorithms above and providing more than 10% energy savings. It can also be used directly after sensors got randomly deployed.

Table 7.1 is a list of summary with our algorithms and some other typical existing algorithms.

The contributions of each of our algorithms are detailed below:

- The Average Relative Position algorithm in Section 4.1 is a low complexity distributed sensor relocation algorithm. It converges more than 2 times faster than the existing solutions with similar computation complexity and also provides some energy savings in the mobilization of sensors. This is achieved by averaging the relative position of sensors. The disadvantage of the algorithm is that the coverage it can reach is 1%-2% less than the algorithms that converge slower.
- The Weighted Relative Distance algorithm in Section 4.2 has a similar convergence rate to the first algorithm we listed above. It only uses the two most important relative distances for calculating the relocation position. Therefore, the error in the localization system can be reduced so that the algorithm can provide enough tolerance to the inaccuracy of the sensor locations. This is very useful in indoor environments which have inaccurate sensor localizations. The disadvantage of the algorithm is that the coverage it can reach is 3% less than the existing solutions.

- The algorithm without location in Section 4.3 uses the relocation energy as a tradeoff, but provides a stable algorithm that increases coverage even without location information. This can reduce the hardware cost of the sensor. However, the travelled distance of sensors will be 20% more than compared to the algorithm using localization systems.
- Our Multi-objective Genetic Algorithm in Section 5.1 is a centralized algorithm with high computation complexity. This will provide more than 40% energy savings in relocating the sensors compared to the distributed solutions. It can provide 100% coverage, while the existing solution in [26] can only provide about 95% coverage.
- We provide two PSO algorithms in Section 5.2 and 5.3 that can use range adjustable sensors to provide energy savings in the WSN. Both can provide significant energy savings while maximizing coverage to 100% of the sensing field. We hybridized our distributed algorithm into the PSO, and the resulting hybrid PSO improved the performance of the PSO by up to 20%.
- A range adjustment algorithm is separately developed for range-adjustable sensors. It can provide 20% - 50% energy savings in sensing. The algorithm can cooperate with all algorithms listed above or scheduling sensor problems, and still provide 100% coverage

Table 7.1 Summary of Algorithms

Algorithm	Section or Reference	C/ D	Location	Major Problem	Complexity	Coverage	Energy	Convergence	Fault tolerant
Average Relative Position	4.1	D	YES	Coverage	Low	95%+	80%	5	NO
Weighted Relative Distance	4.2	D	YES	Coverage	Low	95%	100%	10	YES
Without Localization	4.3	D	NO	Coverage	Low	94%	130%	20	NO
Potential Field	[7][8]	D	YES	Coverage	Low	95%+	100%	20-30	NO
Virtual force	[9]	D	YES	Coverage	Low	93%	110%	20	NO
Voronoi based relocation	[12]	D	YES	Coverage	Medium	95%	80%	10	NO
Multi-objective Genetic Algorithm	5.1	C	YES	Coverage & Moving Energy	High	100%	Less 50%	N/A	N/A
Particle Swarm Optimization	5.2	C	YES	Coverage & Sensing Energy	High	100%	35%	N/A	N/A
Hybrid PSO	5.3	C	YES	Coverage & Sensing Energy	High	100%	30%	N/A	N/A
MOGA by Jourdan	[19]	C	YES	Communication & Coverage	High	N/A	N/A	N/A	N/A
Non-dominated sorting GA	[20]	C	YES	Sensor Scheduling	High	N/A	N/A	N/A	N/A
Limited-mobile WSN optimization	[36]	C	YES	Coverage	High	95%	Less 60%	N/A	N/A
Particle Swarm Gentic Optimization	[39]	C	YES	Coverage	High	6% More	5% mobile	N/A	N/A
Sensing range adjustment	6	D	YES	Coverage & Sensing Energy	Medium	100%	20%-50%	N/A	NO

In conclusion, a complete analysis of coverage optimizations for mobile wireless sensor networks has been made in this thesis. We enhance the energy performance beyond that of existing solutions, even though they produce similar outcomes in optimized coverage. We also addressed the problem of inaccurate localization systems. This is a new way of solving this type of problem and no existing works has addressed this problem. The algorithms derived in the thesis are also self-healing so that they can be used in severe environments as well.

7.2 Future Work

This work focuses on optimizing the deployment of mobile wireless sensors so that coverage and lifetime of the sensor network can be optimized. Our work can be expanded for further research.

Our work focuses on analytical design and produces algorithms tested in the software simulations. Based on the simulations, experimental works can be implemented with the deployment of real mobile wireless sensor networks. For example, we can use the programmable mobile sensor kits and test the relocation algorithms in indoor and outdoor environments. Different hardware modules can be added or removed from the sensor nodes to more closely resemble specific applications.

The algorithms we designed are mainly for monitoring sensors which have specific sensing radii. These algorithms can also be expanded to work with the sensors that collect data at single points. For example, mobile sensors are deployed for monitoring air pollutions. Sometimes it is desirable to redeploy the sensors for an even distribution. In such cases, the distributed optimization algorithms can be applied for deploying the sensors evenly using less energy.

Beyond the sensing coverage problem, the radio frequency coverage is also an active topic in mobile wireless research. Methods for increasing the coverage of sensor networks while maintaining or increasing the radio frequency reception are other ways to proceed from this work.

LIST OF REFERENCES

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and a. J. Anderson, "Wireless sensor networks for habitat monitoring," in *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, Sept. 2002.

[2] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh and a. D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, San Jose, CA, Oct. 2002.

[3] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees and a. M. Welsh, "Monitoring volcanic eruptions with a wireless sensor network," in *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN'05)*, Jan. 2005.

[4] DARPA, "Self-healing Mines," [Online]. Available: http://www.darpa.mil/ato/programs/SHM/.

[5] P.-J. Wan and C.-W. Yi, "Coverage by randomly deployed wireless sensor networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2658-2669, 2006.

[6] B. Wang, K. C. Chua, V. Srinivasan and W. Wang, "Information Coverage in Randomly Deployed Wireless Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 6, no. 8, pp. 2994-3004, 2007.

[7] A. Howard, M. Mataric and G. Sukhatme, "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," in *the 6th International Symposium on Distributed Autmomous Robotics System (DARS)*, Fukuoka, Japan, Jun 25-27, 2002.

[8] W. Sheng, G. Tewolde and S. Ci, "Micro Mobile Robots in Active Sensor Networks: Closing the Loop," in *Proceeding of IEEE International Conference on intelligent Robots and Systems (RSJ)*, Beijing, China, October 9-15, 2006.

[9] Y. Zou and K. Chakrabarty, "Sensor Deployment and Target Localization Based on Virtual Fources," in *IEEE Societies Twenty-Second Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*, San Fransisco, USA, April 1-3, 2003.

[10] S. Li, C. Xu, W. Pan and Y. Pan, "Sensor deployment optimization for detecting maneuvering targets," in *8th International Conference on Information Fusion*, PA, USA, July 25-29, 2005.

[11] T. Wong, T. Tsuchiya and T. Kikuno, "A self-organizing technique for sensor placement in wireless micro-sensor networks," in *18th International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan, March 26-29, 2004.

[12] G. Wang, G. Cao and T. F. Porta, "Movement-Assisted Sensor Deployment," *Transactions on mobile computing*, vol. 5, pp. 640-652, 2006.

[13] M. R. Pac, A. M. Erkmen and I. Erkmen, "Scalable Self-Deployment of Mobile Sensor Networks: A Fluid Dynamics Approach," in *Proceeding of IEEE International Conference on intelligent Robots and Systems (RSJ)*, Beijing, China, October 9-15, 2006.

[14] R.-S. Chang and S.-H. Wang, "Self-Deployment by Density Control in Sensor Networks," *IEEE transactions on vehicular technology*, vol. 57, pp. 1745-1755, 2008.

[15] O'Rourke and Joseph, "Art Gallery Theorems and Algorithms," New York: Oxford University Press, 1987.

[16] Barricell and N. Aall, "Symbiogenetic evolution processes realized by artificial methods," *Methods*, pp. 143-182, 1957.

[17] F. Alex, "Simulation of genetic systems by automatic digital computers. I. Introduction," *Aust. J. Biol. Sci*, vol. 10, pp. 484-491, 1957.

[18] A. Fraser and D. Burnell, Computer Models in Genetics, New York: McGraw-Hill, 1970.

[19] D. B. Jourdan and O. L. d. Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," *IEEE Vehicular Technology Conference*, pp. 2466-2470, Los Angeles, September 26-29, 2004..

[20] J. Jia, J. Chen, G. Chang, J. Li and a. Y. Jia, "Coverage Optimization based on Improved NSGA-II in Wireless Sensor Network," *IEEE International Conference on Integration Technology*, pp. 614-618, 2007.

[21] X. Wang, S. Wang and D. Bi, "Dynamic sensor nodes selection strategy for wireless sensor networks," *International Symposium on Communications and Information Technologies*, pp. 1137-1142, Sydney, Australia, October 16-19, 2007.

[22] R. A. Burne, A. L. Buczak and a. Y.C.Jin, "A self-organizing, cooperative sensor network for remote surveillance:current result," *13th Annua lInternational Symposium on AeroSense Conference–Unattended Ground Sensor Technologies and Applications*, pp. 238-248, Orlando, FL, 1999.

[23] H.-C. Jang and a. H.-C. Lee, "A Voronoi dEtection Range Adjustment (VERA) approach for energy saving of wireless sensor networks," *International Conference on Parallel and Distributed Systems*, vol. 2, pp. 1-7, Taiwan, December 5-7, 2007.

[24] L.-C. Wei, C.-W. Kang and J.-h. Chen, "A force-driven evolutionary approach for multi-objective 3D differentiated sensor network deployment," *IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pp. 983-988, Macao, China, October 12-15, 2009.

[25] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Western Australia, November 27 - December 1, 1995.

[26] R. Kulkarni and G. Venayagamoorthy, "Particle Swarm Optimization in Wireless-Sensor Networks: A Brief Survey," *IEEE Transactions on Systems*, Man, and Cybernetics, Part C: Applications and Reviews, vol. 41, no. 2, pp. 262-267, 2011.

[27] T.-P. Hong and a. G.-N. Shiu, "Allocating multiple base stations under general power consumption by the particle swarm optimization," *IEEE Swarm Intelligence Symposium (SIS)*, pp. 23-28, Honolulu, Hawaii, April 1-5, 2007.

[28] K. S. Low, H. A. Nguyen and a. H. Guo, "Optimization of sensor node locations in a wireless sensor network," *Fourth International Conference on Natural Computation (ICNC)*, vol. 5, p. 286–290, October 18-20, 2008.

[29] H. Guo, K.-S. Low and a. H.-A. Nguyen, "Optimizing the localization of a wireless sensor network in real time based on a low-cost microcontroller," *IEEE Transation on Industrial Electronics*, vol. 58, no. 3, pp. 741-749, 2011.

[30] R. Kulkarni, G. Venayagamoorthy and M. Cheng, "Bio-inspired node localization in wireless sensor networks," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 205-210, San Antonio, Texas, October 11-14, 2009.

[31] K. Veeramachaneni and L. Osadciw, "Swarm intelligence based optimization and control of decentralized serial sensor networks," *IEEE Swarm Intelligence Symposium*, pp. 1-8, St. Louis, Missouri, September 21-23, 2008.

[32] T. Wimalajeewa and S. Jayaweera, "Optimal Power Scheduling for Correlated Data Fusion in Wireless Sensor Networks via Constrained PSO," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3608-3618, 2008.

[33] X. Cao, H. Zhang, J. Shi and G. Cui, "Cluster Heads Election Analysis for Multi-hop Wireless Sensor Networks Based on Weighted Graph and Particle Swarm Optimization," *Fourth International Conference on Natural Computation*, vol. 7, pp. 599 - 603, Jinan, China, October 18-20, 2008.

[34] N. Latiff, C. Tsimenidis and B. Sharif, "Energy-Aware Clustering for Wireless Sensor Networks using Particle Swarm Optimization," *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1-5, Athens, Greece, September 3-7, 2007.

[35] X. Wang, J. Ma, S. Wang and D. Bi, "Distributed Energy Optimization for Target Tracking in Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 73-86.

[36] X. Bai, S. Li, C. Jiang and Z. Gao, "Coverage Optimization in Wireless Mobile Sensor Networks," *5th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1-4, Beijing, China, September 24-26, 2009.

[37] X. Wang, S. Wang and a. J.-J. Ma, "An Improved Co-evolutionary Particle Swarm Optimization for Wireless Sensor Networks with Dynamic Deployment," *Sensors*, vol. 7, p. 354–370, 2007.

[38] Z. Huang and T. Lu, "A particle swarm optimization algorithm for hybrid wireless sensor networks coverage," *IEEE Symposium on Electrical & Electronics Engineering*, pp. 630-632, Kuala Lumpur, Malaysia, June 24-27, 2012.

[39] J. Li, K. Li and a. W. Zhu, "Improving sensing coverage of wireless sensor networks by employing mobile robots," *Proceedings of the International Conference on Robotics and Biomimetics (ROBIO)*, pp. 899–903, Sanya, China, December 15-18, 2007.

[40] N. D and N. B, "DV-based positioning in ad hoc networks," *Telecommunication Systems*, pp. 267-280, 2003.

[41] J. Li, J. Zhang and L. Xiande, "A Weighted DV-Hop Localization Scheme for Wireless Sensor Networks," *Scalable Computing and Communications Eighth International Conference on Embedded Computing*, pp. 269-272, 25-27 Sept Dalian, China, September 25-27, 2009.

[42] Z. Zhao-yang, G. Xu, L. Ya-peng and S.-s. Huang, "DV-Hop Based Self-Adaptive Positioning in Wireless Sensor Networks," in *5th International Conference on Wireless Communications, Networking and Mobile Computing*, Marrakech, Morocco, October 12-14, 2009.

[43] L. Brito, Y. Liu and Y. Garcia, "An improved error localization on DV-Hop scheme for wireless sensors networks," *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE),* vol. 2, pp. 80-84, Chengdu, China, August 20-22, 2010.

[44] C. Xiaoyan, H. Ting and G. Mingjie, "Improved DV-Hop Algorithm Based on Self-Localization of WSN," *1st International Conference on Information Science and Engineering (ICISE)*, pp. 4026-4029, Nanjing, China, December 26-28, 2009.

[45] A. Boukerche, H. Oliveira, E. Nakamura and A. Loureiro, "DV-Loc: a scalable localization protocol using Voronoi diagrams for wireless sensor networks," *IEEE Wireless Communications*, Vols. 6, issue 2, pp. 50-55, 2009.

[46] Y. Wang, X. Wang, D. Wang and D. Agrawal, "Range-Free Localization Using Expected Hop Progress in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vols. 20, issue 10, pp. 1540-1552, 2009.

[47] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," *IEEE Personal Communications*, pp. 28-34, 2000.

[48] K. Xu, M. Chen and Y. Liu, "A Novel Localization Algorithm Based on Received Signal Strength Indicator for Wireless Sensor Networks," *International Conference on Computer Science and Information Technology*, pp. 249-253, Singapore, August 29- September 2, 2008.

[49] X. Li, "Performance study of RSS-based location estimation techniques for wireless sensor networks*," IEEE Military Communications Conference (MILCOM)*, vol. 2, pp. 1064-1068, Atlantic City, NJ, October 17-20, 2005.

[50] X. Li, "5. RSS-Based Location Estimation with Unkown Pathloss Model," *IEEE Transactions on Wireless Communications*, Vols. 5 , Issue: 12 , pp. 3626-3633 , 2006.

[51] J. MacDonald, D. Roberson and D. Ucci, "Location Estimation of Isotropic Transmitters in Wireless Sensor Networks," *IEEE Military Communications Conference (MILCOM)*, pp. 1-5, Washington, DC, October 23-25, 2006.

[52] Q. Shi, H. Huo, T. Fang and R. Li, "Using steepest descent method to refine the node positions in wireless sensor networks," *IET Conference on Wireless, Mobile and Sensor Networks*, pp. 1055-1058, Shanghai, China, December 12-14, 2007.

[53] J. Yao, J. L, L. Wang and Y. Han, "Wireless Sensor Network Localization Based on Improved Particle Swarm Optimization," *International Conference on Computing, Measurement, Control and Sensor Network*, pp. 72-75, Taiyuan, China, July 7-9, 2012.

[54] J. Huanxiang, W. Yong and T. Xiaoling, "Localization algorithm for mobile anchor node based on genetic algorithm in wireless sensor network," *International Conference on Intelligent Computing and Integrated Systems*, pp. 40-44, Guilin, China, October 22-24, 2010.

[55] B. Amutha and M. Ponnavaikko, "Locate: A hybrid localization scheme for wireless sensor networks," *IEEE Symposium on Industrial Electronics & Applications*, vol. 1, pp. 295-300, Kuala Lumpur, Malaysia, October 4-6, 2009.

[56] M. R. Rip and J. M. Hasik, The Precision Revolution: GPS and the Future of Aerial Warfare, Naval Institute Press, 2002.

[57] I. Guvenc and C.-C. Chong, "A Survey on TOA Based Wireless Localization and NLOS Mitigation Techniques," *IEEE Communications Surveys & Tutorials*, Vols. 11, Issue: 3 , pp. 107-124, 2009.

[58] K. Yu, Y. Guo and M. Hedley, "TOA-based distributed localisation with unknown internal delays and clock frequency offsets in wireless sensor networks," *IET Signal Processing*, Vols. 3, Issue 2, pp. 106-118 , 2009.

[59] E. Xu, Z. Ding and S. Dasgupta, "Source Localization in Wireless Sensor Networks From Signal Time-of-Arrival Measurements," *IEEE Transactions on Signal Processing*, vol. 59 Issue: 6 , pp. 2887-2897 , 2011.

[60] Z. Ma and K. Ho, "TOA localization in the presence of random sensor position errors," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 2468-2471, Prague, Czech Republic, May 22-27, 2011.

[61] B. Hood and P. Barooah, "Estimating DoA From Radio-Frequency RSSI Measurements Using an Actuated Reflector," *IEEE Sensors Journal*, Vols. 11, Issue: 2 , pp. 413-417, 2011.

[62] J. Xu, M. Ma and C. L. Law, "AOA Cooperative Position Localization," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1-5, New Orleans, LA, November 30- December 4, 2008.

[63] F.-K. Chan and C.-Y. Wen, "Adaptive AOA/TOA Localization Using Fuzzy Particle Filter for Mobile WSNs," *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pp. 1-5, Budapest, Hungary, May 15-18, 2011.

[64] C. Park, H. Cho, D. Park, S. Cho, J. Park and Y. Lee, "AoA Localization System Design and Implementation Based on Zigbee for Applying Greenhouse," *5th International Conference on Embedded and Multimedia Computing (EMC)*, pp. 1-4, Cebu, Philippines, August 11-13, 2010.

[65] C.-C. Pu and W.-Y. Chung, "Mitigation of Multipath Fading Effects to Improve Indoor RSSI Performance," *IEEE Sensors Journal*, Vols. 8,Issue: 11 , pp. 1884-1886 , 2008.

[66] Y. Wu, J. Hu and Z. Chen, "Radio Map Filter for Sensor Network Indoor Localization Systems," *5th IEEE International Conference on Industrial Informatics*, vol. 1, pp. 63-68, Vienna, Austria, July 23-26, 2007.

[67] W. Chen, T. Mei, L. Sun, Y. Liu, Y. Li, S. Li, H. Liang and M.-H. Meng, "Error analyzing for RSSI-based localization in wireless sensor networks," *7th World Congress on Intelligent Control and Automation,* pp. 2701-2706, Chongqing, China, June 25-27, 2008.

[68] N. Aziz, A. Mohemmed and M. Alias, "A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi diagram," *International Conference on Networking, Sensing and Control*, pp. 602- 607, Okayama City, Japan, March, 26-29, 2009.

[69] S. Meguerdichian, F. Koushanfar, M. Potkonjak and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," *Proceedings. IEEE INFOCOM. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1380-1387, Anchorage, Alaska, April 22-26, 2001.

[70] A. Boukerche and X. Fei, "A Voronoi Approach for Coverage Protocols in Wireless Sensor Networks," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 5190-5194, Washington, DC, November 26-30, 2007.

[71] J. Li, R.-c. Wang, H.-p. Huang and L.-j. Sun, "Voronoi Based Area Coverage Optimization for Directional Sensor Networks," *Second International Symposium on Electronic Commerce and Security*, vol. 1, pp. 488-493, Nanchang, China, May 22-24, 2009.

[72] J. Wu and S. Yang, "Coverage issue in sensor networks with adjustable ranges," *Proceedings on International Conference on Parallel Processing Workshops (ICPP)*, pp. 61-68, Montreal, Quebec, Canada, August 15-18, 2004.

[73] M. Cardei, J. Wu, M. Lu and M. Pervaiz, "Maximum network lifetime in wireless sensor networks with adjustable sensing ranges," *IEEE International Conference on Wireless And Mobile Computing, Networking And Communications (WiMob'2005)*, vol. 3, pp. 348-445, Montreal, Canada, August 22-24, 2005.

[74] A. Dhawan, C. Vu, A. Zelikovsky, Y. Li and S. Prasad, "Maximum Lifetime of Sensor Networks with Adjustable Sensing Range," *Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing,* pp. 285-289, Las Vegas, Nevada, June 19-20, 2006.

[75] Z. Pei, Z. Deng, B. Yang and X. Cheng, "Application-oriented wireless sensor network communication protocols and hardware platforms: A survey," *IEEE International Conference on Industrial Technology,* pp. 1-6, Chengdu, China, April 21-24, 2008.

[76] M. Stojcev, M. Kosanovic and L. Golubovic, "Power management and energy harvesting techniques for wireless sensor nodes," *9th International Conference on Telecommunication in Modern Satellite, Cable, and Broadcasting Services (TELSIKS)*, pp. 65-72, October 7-9, 2009.

[77] Y. Mei, Y.-H. Lu, Y. Hu and C. Lee, "Energy-efficient motion planning for mobile robots," *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, vol. 5, pp. 4344-4349, New Orleans, LA, April 26- May 1, 2004.

[78] X. Fei, A. Boukerche and R. Araujo, "Irregular Sensing Range Detection Model for Coverage Based Protocols in Wireless Sensor Networks," *IEEE Global Telecommunications Conference (GLOBECOM)*, pp. 1-6, Honolulu, Hawaii, Nov. 30 2009-Dec. 4 2009.

[79] R. Eckhardt, "Stan Ulam, John von Neumann, and the Monte Carlo method," *Los Alamos Science*, Special Issue (15), p. 131–137, 1987.

[80] M. H. Kalos and P. A. Whitlock, Monte Carlo Methods, Wiley-VCH, ISBN 978-3-527-40760-6, 2008.

[81] D. W. Boll, J. Donovan, R. L. Graham and B. D. Lubachevsky, "Improving Dense Packings of Equal Disks in a Square," *Electronic Journal of Combinatorics*, vol. 7, 2000.

[82] X. Wang, F. Sun and X. Kong, "Research on Optimal Coverage Problem of Wireless Sensor Networks," in *WRI International Conference on Communications and Mobile Computing*, Kunming, China, January 6-8, 2009, pp: 548-551.

[83] J. C. Andreas, Energy-Efficient Electric Motors, Marcel Dekker, 2nd Edition, 1992.

[84] P. Bertoldi and A. T. d. Almeida, Energy Efficiency Improvements in Electronic Motors and Drives, Springer, ISBN: 978-3540674894, June 22, 2000.

[85] Grigoriy B. Reshko. Matthew T. Mason and Illah R. Nourbakhsh, "Rapid Prototyping of Small Robots," Technical report, CMU-RI-TR-02-11, Camegie Mellon University, 2002.

[86] Y. Mei, Y.-H. Lu, Y. Hu and C. Lee, "Energy-efficient motion planning for mobile robots," in *Proceedings of IEEE International Conference on Robotics*

and Automation (ICRA), New Orleans, LA, USA, New Orleans, LA, USA, April 25- May 1, 2004, vol 5, pp 4344- 4349.

[87] A. Dhawan, C. Vu, A. Zelikovsky, Y. Li and S. Prasad, "Maximum Lifetime of Sensor Networks with Adjustable Sensing Range," in Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, Las Vagas, USA, June 19-20, 2006, pp: 285 - 289.

[88] Texas Instruments, [Online]. Available: http://www.ti.com/tool/cc4000gpsem.

[89] Herbert Taub and Donald L. Schilling, Principles of Communication Systems, New York: McGraw-Hill, 1971.

VITA

YIPENG QU

1985	Born, Wuhan, China
2007	B. E., Electronic & Information Engineering Huazhong University of Science & Technology Wuhan, China
2009	M. S., Electronic & Information Engineering Huazhong University of Science & Technology Wuhan, China
2013	Ph. D. Candidate, Electrical Engineering Florida International University Miami, USA

PUBLICATIONS AND PRESENTATIONS

JOURNALS

[1] Yipeng Qu, Stavros V. Georgakopoulos, "An Average Distance Based Self-Relocation and Self-Healing Algorithm for Mobile Sensor Networks", *Journal of Wireless Sensor Networks*, *Vol. 4 No. 11, pp. 257-263, 2012.*

CONFERENCES

[1] Yipeng Qu, Stavros V. Georgakopoulos, "A Distributed Self-Relocating Algorithm for Randomly Deployed Mobile Wireless Sensors", *IEEE Conference on Wireless Sensors and Sensor Networks, Austin, TX, Jan. 20-23, 2013*.

[2] Yipeng Qu, Stavros V. Georgakopoulos, "A Centralized Algorithm for Prolonging the Lifetime of Wireless Sensor Networks using Particle Swarm Optimization", *13th Annual IEEE Wireless and Microwave Technology Conference, Cocoa Beach, FL, Apr. 16-17, 2012, pp: 1-6.*

[3] Yipeng Qu, Stavros V. Georgakopoulos, “Relocation of Wireless Sensor Network Nodes Using a Genetic Algorithm”, *12th Annual IEEE Wireless and Microwave Technology Conference, Clearwater, FL, Apr.18-19, 2011, pp: 1-5.*