Localized Ant Colony of Robots for Redeployment in Wireless Sensor Networks

by

Yuan Wang

Thesis submitted to the

Faculty of Graduate and Postgraduate Studies

In partial fulfillment of the requirements

For Master of Applied Science degree in

Electrical Engineering

School of Electrical Engineering & Computer Science

Faculty of Engineering

University of Ottawa

Acknowledgement

I would give my sincerest gratitude to my supervisor Prof. Amiya Nayak and my co-supervisor Prof. Ivan Stojmenovic for the exceptional guidance and support through all my graduate studies, not only for this thesis but also the research work. Without their instructions, I cannot learn so much and complete this thesis. I am grateful to Dr. Cheng Wang who gave me valuable advices and suggestions.

I wish to express my cheers to my friend, Haotian Li, who gave me encouragement and help during my studies. We will be friends as we were before.

Finally I would like to thank my parents and grandparents who always understand and support my ideas and decisions, this work is dedicated to them.

Abstract

Sensor failures or oversupply in wireless sensor networks (WSNs), especially initial random deployment, create both spare sensors (whose area is fully covered by other sensors) and sensing holes. We envision a team of robots to relocate sensors and improve their area coverage. Existing algorithms, including centralized ones and the only localized G-R3S2[16], move only spare sensors and have limited improvement because non-spare sensors, with area coverage mostly overlapped by neighbour sensors, are not moved, and additional sensors are deployed to fill existing holes. We propose a localized algorithm, called *Localized Ant-based Sensor Relocation Algorithm with Greedy Walk* (LASR-G), where each robot may carry at most one sensor and makes decision that depends only on locally detected information. In LASR-G, each robot calculates corresponding pickup or dropping probability, and relocates sensor with currently low coverage contribution to another location where sensing hole would be significantly reduced. The basic algorithm optimizes only area coverage, while modified algorithm includes also the cost of robot movement. We compare LASR-G with G-R3S2, and examine both single robot and multi robots scenarios. The simulation results show the advantages of LASR-G over G-R3S2.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

A wireless sensor network (WSN) [1] is collection of a number of sensors. Sensors are deployed within a region of interest (ROI) to monitor the surrounding area in real-time environmental conditions. WSNs gained attention after the development of Micro-Electro-Mechanical Systems technology (MEMS), which contributed to the high-tech design of smart sensors [33]. For a WSN in which sensors are initially deterministically deployed, although original deployment can already reach full coverage as [11][17], high computational ability and large energy storage are needed for actuators (robots for simplicity), which are set to execute the deployment of sensors. In this case, random sensor node dropping is preferred with no central control of placement of all sensors at initial status for reducing the cost. Then relocation of sensors happens to replace and increase the coverage over the ROI.

1.1.1 What is a WSN?

A WSN consists of massive micro-sized sensor to monitor physical conditions of nearby area, such as pressure, temperature, sound, etc. It works by the cooperation among sensors, which are converters for measuring physical quantity and converting to signals that can be read by certain electronic instrument. Sensors monitor ROI and when an event happens, at least one sensor will detect it by changing of signals received. Sensor will relay the information of the event to a destination designed to assemble such information updates.

Sensor nodes are self-organized and inexpensive sensing devices with limited computational abilities and storage. They communicate with each other using wireless links in order to perform distributed sensing tasks. Sensors are expected to cover the whole ROI through network's lifetime, thus providing a quality-guaranteed monitoring service. There are two types of sensors: static sensors and mobile sensors. Static sensors obtain the basic sensing abilities within sensing range, while mobile sensors have locomotion with a vehicle device equipped to each sensor. For those static sensors, although cost is reduced, robots need to be involved for possible relocation process. In most WSNs, static sensors are preferred since it is less cost consuming compared with the scenarios using mobile sensors. However, by applying static sensors, robots or other actuators are required to carry and deploy sensors.

For a sensor, sensing range is predefined which is a circle-shaped area with sensor itself as centre of the circle, see in Figure 1.1. Area within a sensor's sensing range can be monitored by this sensor. We call the distance between a sensor and the farther sensing-enable location sensing radius r_s, which is defined as the radius of circle-shaped sensing range. Any location outside sensor's sensing range can no longer be monitored by this sensor. Thus, this specific sensor can only detect events happening

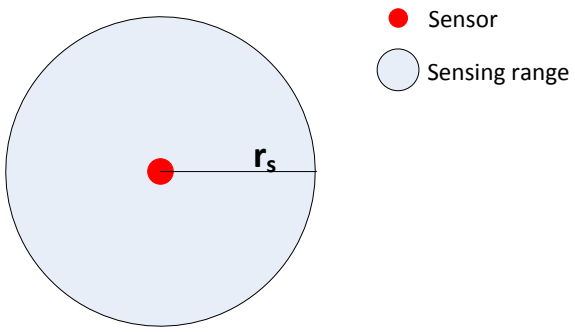


Figure 1.1: Sensor and sensing range.

within sensing range.

Apart from sensing radius r_s, sensor communication radius r_c is defined for each sensor. When a sensor is about to send a message, the farthest distance it can reach is called communication radius r_c. The area with the cycle-shaped area in Figure 1.2(a) is defined as communication range for the sensor, with itself as centre of cycle, r_c as radius. Based on above definitions, any node located within a sensor's communication range is able to received flooding messages from the sender. For any two sensors s_i and s_j, each is located in the other's communication range, we define s_i and s_j as neighbour sensors, see in Figure 1.2(b).

A sensor's sensitivity indicates the percentage of changes the output makes when measured quality changes. For different tasks, sensitivity may vary due to the range the measured input can have. With sensitivity getting higher, more cost is needed for the physical design of the sensors. High sensitivity is necessary when measuring small changes, while low sensitivity for large range of changes. Apart from sensitivity, there are other factors, such as computational abilities, storage and battery, that act as significant bottlenecks for a sensor to improve the performance. Because of the micro-sized instrument designed, memory storage and battery capacity are limited even though it is fully charged. For reducing the cost, computational abilities are

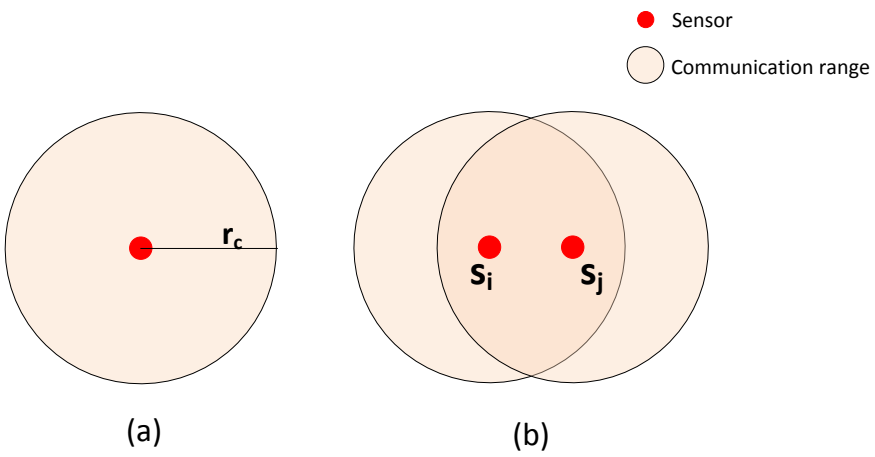


Figure 1.2: Definition of communication range and neighbour sensors.

also confined. Thus, in order to make sensors work functionally for a longer lifetime, some strategies, such as relocation of the deployed sensors, are definitely essential for WSNs and sensor connectivity maintenance purposes.

1.1.2 Why Relocation

There are two initial ways to construct a WSN, which are deterministic deployment and random deployment of sensors. When applying deterministic deployment, sensors are always left at a best location with minimum possible coverage overlap with other nearby sensors. Static sensors are carried by robots and dropped along with robots movements [2][17]. In random deployment, sensor nodes are dropped stochastically within ROI and relocation some targeted sensors based on predefined principles. Through relocation process, coverage is increased since overlay of sensing range among sensor is reduced.

By deterministic deployment, coverage enhancement can be guaranteed since robots will calculate topological information and optimize deployment location. However it will introduce long time delay for deployment process since robots have to go

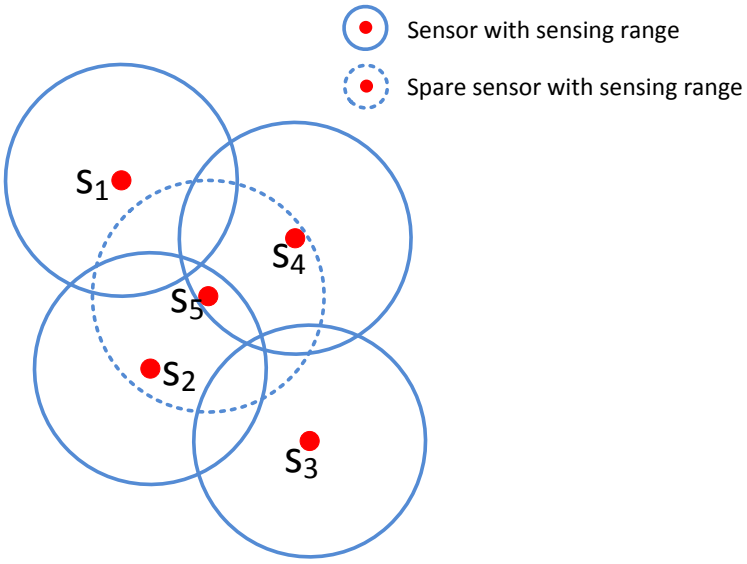


Figure 1.3: An example for defining a spare sensor.

through the whole ROI, which may even cause sensor node failure because of limited battery storage even if fully charged. It is possible that the first deployed sensor node runs out of battery before whole deployment procedure finishes. Also, too much calculation is involved to robots since it needs to determine the location for every sensor. Though it can easily come with a good deployment solution, time delay and computational burden for robots are influential to the performance of deployed WSN, especially for real-time situation.

Due to the shortcomings of deterministic deployment mentioned above, sensor relocation protocols with initial random deployment are more preferred for runtime task completion. At initialization, sensor nodes are dropped stochastically in ROI without any control of location guidance. By random node dropping, computation and time delay issues are improved. However spare sensors and sensing holes may created for random placement. We define a sensor as a spare sensor if its own sensing range is fully covered by other sensors. As shown in Figure 1.3, sensing range of

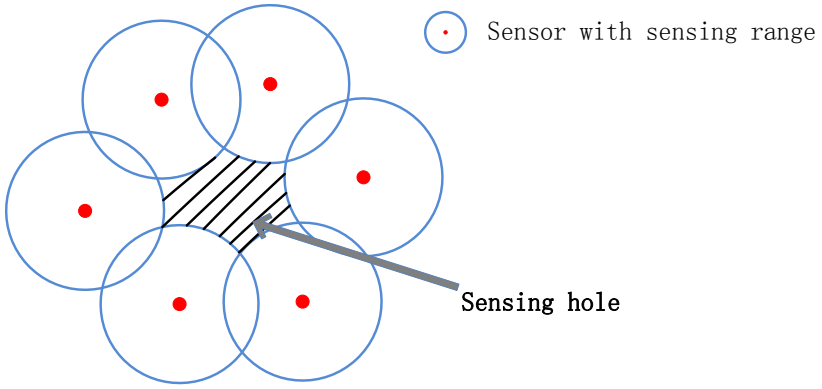


Figure 1.4: An example for defining a sensing hole.

sensor s_5 is totally overlapped by sensors from s_1 to s_4. When a sensor finds itself a spare sensor, it turns to "sleeping" mode and does not execute any monitoring tasks. Sensors whose sensing range is not totally covered by other neighbour sensors will stay active for WSN connectivity. Sensing holes are polygon-shaped areas that are not monitored by any sensor (example in Figure 1.4), which means not within any sensor's sensing range. After random deployment, relocation protocols are applied to move targeted sensors (spare sensors or other sensors depending on the protocol) and cover sensing holes by targeted sensors. Spare sensors will "wake up" when being redeployed because it loses connection with neighbour sensors and because it is no longer fully overlapped with neighbour sensors. Relocation of targeted sensors will improve coverage performance in ROI since it reduces the overlapping of sensing range among sensors.

1.1.3 Actuators in Sensor Relocation

An actuator is a type of motor for moving or controlling a mechanism or a system. It works with some kind of supported energy, such as electric current, and converts energy source to motions. It acts as the mechanism and executes tasks for a control

system.

In a WSN, actuators (robots for simplification) may be needed for sensor relocation depending on the properties of deployed sensors. If sensors are mobile with locomotion, it can self-deploy to another location without the help of robots. However, since static sensors are more preferred for saving sensor energy consumption, robots are involved to relocation in those scenarios. Robots act as vehicles to pick up sensors, carry them while moving and drop at different locations. Because of the size and battery storage, robots can store more energy compared with sensors. Thus, it saves sensors energy since sensors are carried by robots. When robots are about to run out of energy while relocating sensors, it will stop the tasks and walk directly to a base station for energy recharging. When fully charged, it will walk back to events based on some preset walking principles.

Similar to sensors, robots also have a preset value for communication radius R_c. Sensors in robot communication range are able to receive messages sent from robot. However, if a robot is able to send a message to a sensor, it does not guarantee robot will receive a reply from the sensor. This is because that in most situations, $R_c > r_c$, even if a sensor is located in robot's communication range, it is possible robot is outside sensor's communication range, see in Figure 1.5.

For each robot a, it can carry limited number of sensors in relocation process. The maximal number of sensors robot a can carry is N_c. When a reaches its maximal cargo N_c, it has to first drop at least a sensor before picking up other ones. Similar, it needs to pick up sensors first when it carries no sensors at current location.

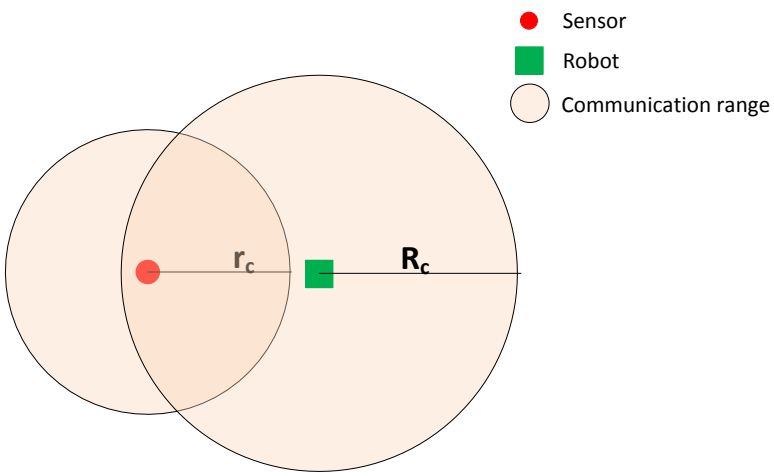


Figure 1.5: Definition of communication range and neighbour sensors.

1.2 Problem Statement

The main purpose for sensor relocation protocols is to increase the sensing coverage compared to original deployment. In order to increase the performance for WSNs after initial deployment, methods for relocation of sensor nodes are introduced recently. We consider a WSN which is constructed by stochastically dropping sensor in ROI in the initial state. Sensors communicate with others in communication range in order to find neighbour sensors and identify spare sensors. Relocation methods are applied to redeploy some of the sensors and reduce overlay among sensing range among neighbour sensors.

Relocation can be beneficial for both improving the coverage by recovering sensing holes within ROI after initial random deployment [12][16], and replacing failed sensor nodes after running out of battery [9][19][32]. For the former, sensors are redeployed to cover at least part of sensing holes according to some relocation methods, while the latter focuses on dropping spare sensors directly on the exact same location of failed sensors since the original WSN before failure may already be well deployed.

Before making any decisions about choosing targeted sensors or finding a best destinations for the improvements, topological information should be gained by sensors, and possible robots (for static sensors), such as location of sensors and robots, if obstacles exist in ROI, boundaries of ROI, etc. Two common assumptions can be made for grasping information, which are classified as centralized methods and localized methods. For centralized methods (e.g. [9][28]), it is assumed that all of topological information is pre-known to sensors and robots. It works by relaying every local information to a base station, and some specific nodes can obtain useful information from base station. Through centralized information grasping, a global best solution for spare sensor and sensing holes detection as well as route selection is calculated even before any actions and optimizations are made for a ROI. In localized methods such as [19][16], local information is not shared to all nodes by routing. Information is all exchangeable among nodes within communication range. Thus a node can only get information from its 1-hop neighbours. In that case, sensors and robots calculate and make local best decisions based on local information. Although a global best solution for coverage enhancement can be obtained in centralized methods, the cost for message relay is higher than localized methods. Besides, high computational ability is required for base station since it needs to manage all the data related to ROI. And sensors located near base station will have heavier burden on message transmission, which will cost more energy consumption when comparing with the ones with less relay. In this way, these sensors will create a bottleneck for the WSN because it is more likely to suffer from node failure by running out of battery. Thus, choosing from these two assumptions is a tradeoff between performance of coverage improvement and total energy consumption.

Usually relocation methods take plenty of rounds by repeating redeployment of targeted sensors. Stopping criterion is set by users and is different for each situation

based on the detailed parameter desired to be improved. For simplification purposes, it is set when no targeted sensor is detected for a preset time window, or a preset processing time is reached.

1.3 Motivation

In a WSN, if the sensing area of a sensor is fully covered by other neighbour sensors, this sensor should turn to "sleep" mode and label itself as spare sensor for saving battery. Most existing algorithms related to sensor relocation focus on moving spare sensors to an uncovered area, in order to "wake up spare sensors" and increase total coverage. However, if the number of initial deployed sensors is limited in a ROI, it may be possible that large sensing holes still exist even after all spare sensors are relocated and start to monitor a part of subarea within the sensing range, if we assume no extra sensors are added to ROI in relocation process. In that case, improvement of coverage is limited by the number of spare sensors and original distribution among sensors in ROI. If extra sensors are allowed to fix sensing holes, energy cost is increased for those extra sensor.

In self-redeployment algorithms (e.g. [19][34]), mobile sensors detect spare sensors and sensing holes by message relay. Then spare sensors move to cover sensing holes according to some preset principles. However, by moving directly to a sensing hole area, a mobile sensor may lose energy for travelling from the original place to a destination. It may even happen that after it travels a long distance to the destination, it loses most stored energy and is about to face a sensor node failure. Besides, by travelling for a long distance, travelling time delay will prolong the relocation process. In this case, existing algorithms will improve relocation methods as shown in Figure 1.6. Sensor s_1 is a targeted sensor detected for relocation process. Its

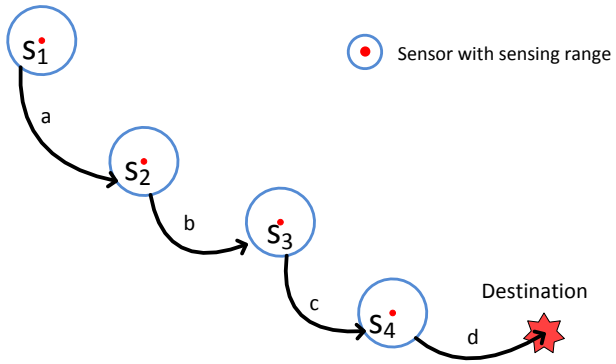


Figure 1.6: An example of how mobile sensors move to destination.

destination is a sensing hole mark red in the figure. Instead of moving directly to the destination, s_1 finds a neighbour sensor s_2 whose location is in the similar direction to destination, and moves to s_2. At the same time when s_1 moves a is happening, b, c and d rephrase a sensor in source to next sensor's location. These simultaneously movements will decrease repair time and save energy for original targeted sensor. However it may cause disconnection among neighbour sensors during relocation. Due to above reasons, we will focus on using static sensors and robots to deal with sensor relocation issues.

With robot-assisted methods, there is no need for sensors to accomplish all of the computation. The reduction of computational ability of sensors can therefore lead to less hardware investment and memory storage. Due to the physical differences for designing sensors and robots, micro-sized sensors contain far less energy even if fully charged compared to robots. With robots undertaking more computation, sensors' lifetime is prolonged, which is one of the most crucial criteria for WSN performance maintenance.

Considering some existing robot-assisted sensor deployment schemes [12] [11] [9], robots detect spare sensors and sensing holes according to some principles. Then

robots make decisions and move spare sensors to cover sensing holes. For some situation when sensing holes still exist but no more spare sensors is detected by robots, a base station is needed in which enough spare sensors are stored to recover ROI. However, by adding extra sensors, energy usage will be increased compared with relocating existing active sensors, since the latter one will reduce overlapping of sensing area among active sensors. Except for storing extra spare sensors, a base station is also used to grasp global topological information within ROI in centralized algorithms [9] [10], which result in heavy computational burden for base station. Thus, these existing algorithms can only come to a good performance under very limited conditions.

In [16], coverage is optimized by an algorithm named *Randomized Robot-assisted Relocation of Static Sensors* with a grid-based variant (G-R3S2). In G-R3S2, sensors check coverage area, label as spare sensors and inform neighbour proxy sensors if fully covered by other active sensors. Each sensor identifies the arcs on its sensing range periphery that are not covered by any other active sensor. The uncovered arcs are part of boundaries of polygon-shaped sensing holes. Coverage is improved by robots relocating spare sensors to cover nearby sensing holes. Robot random movement is restricted by walking on virtual grid and applying a *Least Recently Visited* policy (LRV), which indicates robot to move to a least recently visited adjacent grid point. However, performance of G-R3S2 is determined by a number of sensors deployed in ROI. With fewer sensors involved, it only get limited increment on coverage.

In biological research about ants movements, we find that ants clean their nests and organize dead bodies in their colonies by clustering based on their properties. Data-clustering algorithms[31][6] are proposed earlier on the basis of ant colony clustering objects. The goal of ant-based data clustering is to import original data into a bi-dimensional output grid and grouping items that are similar to each other in their

attributes in neighbour regions of the grid. In these clustering algorithms, ant-based agents are set into the grid for picking up and dropping off data objects according to some principles for clustering unlabelled datasets. Although designed for different goals, we think about proposing a relocation protocol acting as an anti-clustering methods inspired by ant-based data clustering algorithms.

1.4 Contributions

We present a new approach to address the problem of sensor replacement using mobile robots, named *Localized Ant-based Sensor Relocation with Random Walk* (LASR-R), and improved it with *Greedy Walk* (LASR-G). These algorithms aim at spreading sensors for coverage improvement with fewer sensors deployed in ROI. This approach is innovative not only because it is the first localized algorithm considering the movement of active sensors, but also it is inspired by ant-based algorithms for data clustering [31] [6], which is the opposite goal from balancing the placement of sensors within ROI.

We consider that robots can pick up both the spare sensors and some active sensors under a certain criterion, although with spare sensors gaining higher priority. While taking each step, robots calculate the corresponding picking up or dropping probability and make decisions after collecting location information of neighbour sensors within the sensor's communication range. Decisions are made according to local information obtained by robots from neighbour sensors. After a robot either picks up a sensor or drops one, it will walk along a greedy direction a step away from the current location. In LASR-R, robots are designed to take a random walk to an direction within ROI. Step length is preset by users. For LASR-G, robots are guided by neighbour sensors to choose a local best step. Unlike most existing algorithms,

LASR-G contains low message overflow since no central information is needed. Sensors are also not involved in much calculation and message routing, which can prolong sensor's lifetime and keep WSN work with less node failure. During simulation work, we compare the performance of single-robot and multi-robots scenarios, and calculate the coverage percentage and average traveling distance for robots.

1.5 Organization

This thesis is organized as follows:

- Chapter 2 is a background introduction about sensor relocation protocols in WSN and ant-based data clustering methods. This chapter presents existing related work about sensor relocation, compares and analyzes limitations about these algorithms. A brief description of ant-based data clustering methods, which is the inspiration of our proposed algorithms, is described in this chapter. We discuss the original goal of data clustering methods and how it applies to sensor relocation issues.

- Chapter 3 introduces the concept of robot-assisted sensor relocation and coverage ratio. A novel localized relocation method is presented in this chapter, with two algorithms focusing on different relocation issues. Proposed method is the first one to consider relocation of active sensors to improve coverage. Robot path selection is considered for proposed algorithms as a combination of greedy walk and random walk.

- In Chapter 4, we discuss the parameter measurement for the proposed algorithms. From the evaluation of tested merits for both our two proposed algorithms and a competing algorithm, it is proved that the proposed algorithms

outperform existing algorithms on coverage improvement and robot traveling distance. It is also proved that by relocation of some active sensors, robot takes less moves to search for targeted picking up sensors. For each of the two proposed algorithm, the simulation results and protocol comparison illustrate that it outperforms than other algorithms on its own relocation issue.

- Chapter 5 discusses conclusion and future work.

Chapter 2

Literature Review

Normally there are two way of deployment: deterministic deployment and random deployment. However, in most cases we prefer the latter one since it can both save energy and reduce cost. For random deployment, sensor relocation is needed to move some targeted sensors to better locations considering the coverage performance and some other measured properties. Some methods about sensor relocation have been proposed these years. Sensor relocation protocols are relevant as a fault-tolerance approach to maintain sensing coverage[19], at the same time prolonging lifetime of wireless sensor network and maintaining topology unchanged[18]. Normally there are two types of algorithm: centralized algorithms and localized algorithms. In centralized algorithms, central nodes are usually needed to collect and record network information. In localized algorithms, sensors only communicate with their neighbors in most cases. Although centralized algorithms are more likely to lead to a more accurate solution, large computation and message complexity may affect the performance in the end. In contrast, localized algorithms are scalable and will reduce the time and energy for computation.

2.1 Deterministic Sensor Deployment in WSNs

Some existing sensor deployment algorithms are proposed, aiming to distribute sensors to construct a WSN. Location for each sensor is calculated either by actuators in robot-assisted algorithms, or sensor itself if locomotion is attached to it. After deployed, a WSN is constructed with the whole ROI monitored by well deployed sensors.

Batalin et al. [2] proposed an algorithm for sensor deployment with single robot. Robot takes random move and place sensors if current location is not occupied by other sensors. Sensors record directions and movements of robot, and then give suggestions to robot about route selection if within communication range. *Least Recently Visited* (LRV) policy is introduced to direct robot to a least recently visited place. Sensors count times of robot walking to each of four directions and update the numbers in memory. Every time robot arrives a sensor's location, it asks sensor for direction information of next step.

A back-tracking algorithm for multi-robots sensor deployment [11] was presented. Sensors are deployed on the basis of virtual grid according to a pre-defined direction priorities. While a robot is walking along virtual grids, it memorizes deployed status of area in vicinity at each location. At location a, if robot detects uncovered area in neighbourhood, it records this uncovered area and keeps walking in same direction along grid. If obstacles are met, robots track back to the last visited location recorded to have connection to one or more uncover sensing holes.

Deterministic deployment methods tend to get well deployed WSNs when task finished, because it optimizes sensor deployment at each location. However with deterministic deployment methods involved, relocation time is prolonged, which will create larger time delay for real-time WSN monitoring system.

2.2 Sensor Self-deployment Algorithms

In sensor self-deployment algorithms, all sensors are mobile with locomotion. Actuators are not necessary for relocation purposes. With the use of mobile sensors, failed sensors or sensing holes can be directly replaced if spare sensors can detect coverage holes and walk towards them. However, due to massive number of sensors involved in a WSN, cost for applying mobile sensors will increase comparing with static sensors.

2.2.1 Biologically Inspired Relocation Approach

Several recent biologically inspired (particle swarm optimization, genetic, ant colony) algorithms [21] [28] [29] [22] assume that sensors are mobile, and able to search and relocate themselves based on information available on area coverage. These algorithms are centralized and cannot be easily converted to a distributed solution. The main issue is that such solutions require global information on the network status, or local information that does not exist in the field. For example, several ant colony based algorithms assume that sensors leave pheromone traces in the field in places where no sensor exists, and finds traces from other sensors on some edges between virtual nodes. Swarm intelligence was applied to other problems in sensor networks, such as intrusion detection [15].

2.2.1.1 Particle Swarm Intelligence Based Approach

A particle swarm optimization (PSO) based algorithm was proposed by Qu et al.[28] for sensor relocation. Under this algorithm, both of the sensing radius and traveling distance are optimized for energy saving purposes. Voronoi diagrams are applied to ensure the sensor coverage in the whole network and to help adjust the sensing range for each active sensor.

PSO was first introduced by Kennedy et al.[27]. The idea of PSO is the simulation process of a group of birds searching for a piece of food. It is easier to program and implement, and has less parameters to control. The author assumes the sensors have adjustable sensing range. Among all the assumptions made by the author, one of them is worth the mention: a centre node which have powerful computation ability and unlimited energy resource is needed. Similar to most centralized algorithms, this centre node is in charge of collecting location information for all deployed sensors and run an algorithm to optimize whole lifetime of the sensor network. In this algorithm, the author concerns more about energy issues, especially about the bottleneck, which indicates to the sensor that drains its battery fastest. With the description of fitness function about total energy consumption, energy is spent mostly on sensor movement and sensing tasks. The author uses PSO to minimize the fitness function. Two types of best fitness value, global best and personal best, are considered for updating velocity and position.

Although PSO based algorithm can come to a globalized outcome meanwhile reducing the energy spent for sensor relocation, it also has some drawbacks. Defined as a centralized algorithm, the centre node can be regarded as a base station for collecting information and fulfilling computation tasks. However, the processing speed of centre node is limited, especially for the scenarios which contains massive number of sensors within ROI. It may cause delay or even traffic jam for communicating and flooding messages between sensors and centre node. Besides, the energy consumption of centre node is simply ignored in this paper, which can be one of the main consumption in WSN. With larger energy consumption spent in the whole relocation process, higher cost of WSN will be introduced. Adjustable sensing range may also increase the cost for each micro-sized sensor, and threshold of the sensing range should be considered, especially for sensor networks with low density.

2.2.1.2 Genetic Based Approach

Multi-objective Genetic Algorithm was presented by Qu et al.[29] with the idea derived from heuristic genetic algorithms. Powerful processor is need at base station collecting location information of each sensor. By applying Multi-objective Genetic Algorithm, not only the effect of obstacles is considered, but also both the final positions and travelled distance are optimized with least energy consumption. With the control of mutation ratio, the best value will be reached after a number of generations. In this algorithm, total sensing field is partitioned into a grid for the convenience of judging whether covered by sensors or not. Obstacles can be detected whether blocking sensors by this method. With the use of powerful base station, energy consumption for collecting global information is increased.

2.2.1.3 Ant-Colony Based Approach

Ant Colony Optimization (ACO) was introduced by Dorigo et al.[8]. ACO algorithm is inspired by colonies of real ants, which deposit chemical substance (pheromone). With this imitation of ant behavior, favorable path can be found and other members of the colony can follow this path for object transmission. Ants tend to follow the paths where pheromone concentration is higher.

Usually there are two ways of defining the construction graph that consists of a set of vertices and a set of edges. By graph construction pheromone is deposited either on edges of the graph or on vertices. In ACO algorithms, the author defined solution component, with which the pheromone is associated. At each construction step, partial solutions are extended until the stopping criterion is activated. At the same time, pheromone value is updated by increasing pheromone level with "good" solutions by placing extra pheromone and decreasing with "bad" solutions by evaporation

of pheromone component. Pheromone evaporation is an effective way for the ACO algorithm to avoid too rapid convergence towards a local optimal solution. There are many specific ACO algorithms aiming at solving different problems, especially for NP-hard problems.

Algorithm 1 The Ant Colony Optimization Metaheuristic

```
Set parameters, initialize pheromone trails
while terminating conditions not met do
  ConstructAntSolutions
  ApplyLocalSearch (optional)
  UpdatePheromones
end while
```

Ant Colony Optimization (ACO) based algorithm for sensor deployment was proposed by Liao et al.[21]. The author achieves the goal by modeling sensor deployment as the multiple knapsack problem (MKP)[4]. Based on descriptions in the paper, sensors closer to the sink tend to have higher energy consumption than those far away from the sink. Thus, non-uniformly deployment is needed in ROI to save energy for each sensor.

In the sensor layout model shown in Figure 2.1, as least one sensor must be deployed in the predetermined circle point. One of the sensors in each circle point is selected as the sensor head. The sensor head collects information in circle point and exchange with neighboring sensor heads. As the author indicated, the sensor redeployment problem can be defined as the problem of distributing and balancing the energy among a centre circle point and its neighboring circle points.

In ACO based algorithm, pheromone model is used to find the shortest paths. The pheromone model is a parameterized probabilistic model that consists of a set of model parameters. By accomplishing ant colony algorithm, sensors moving to neighboring circle point should be determined, as well as demand energy and pheromone trails

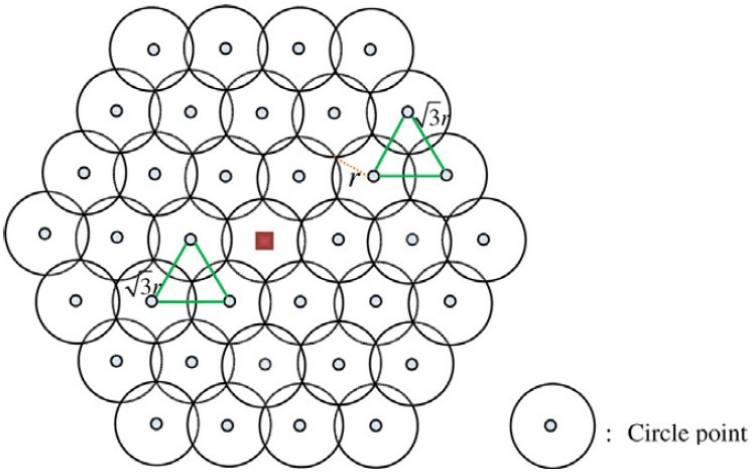


Figure 2.1: The sensor layout model.

updated. For demand energy, if it increases for a circle point, the probability of movement of sensor also increases. For pheromone trails, pheromone level is updated in pheromone table. Evaporation and new addition pheromone are considered in this paper, with setting evaporation parameter and defining new addition pheromone as new lifetime minus original lifetime. However, with the use of mobile sensors and deterministic sensor node dropping, cost of constructing a WSN is higher than using robot-assisted sensor relocation for ROI.

2.2.2 Mesh-Based Approach

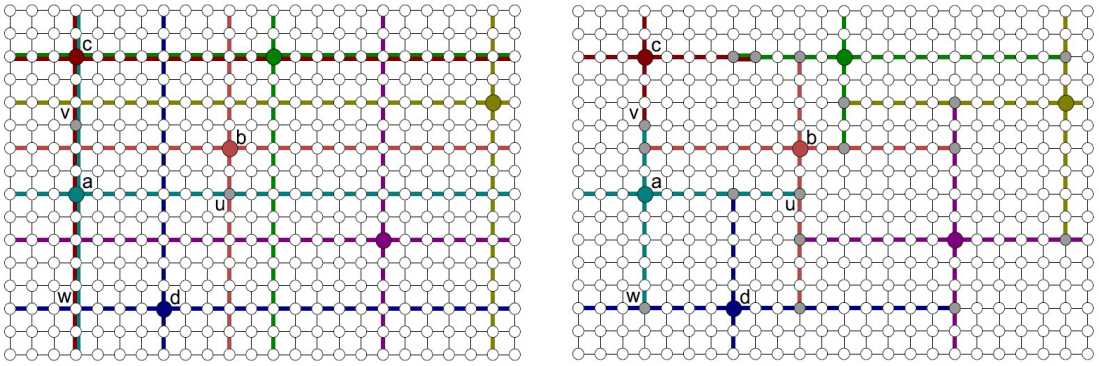
Li et al. introduced information mesh, which is short for iMesh[20], for local data transmission. The authors then proposed a Mesh-based Sensor Relocation Protocol (MSRP)[19] for mobile sensor networks. MSRP maintains sensing coverage by replacing failed sensors with nearby redundant sensors with minimal time delay and balanced energy consumption, with the guidance of iMesh. MSRP protocol works by a shifted node relocation method. By shifting positions of all nodes along a path built between a failed active node (A-node) and a redundant node (R-node), A-node

can be finally replaced by another active sensor as indicated in Figure 1.6.

MSRP is a localized preknowledge-independent algorithm. It aims at shifted node discovery and node replacing with constant relocation delay and balanced energy consumption, especially in those scenarios where large number of randomly scattered R-nodes within a given ROI. In the network topology, location information of an R-node is stored in a proxy node (one of active neighbor sensors of R-node that construct information mesh). In that case, if an event happens exactly at location of a proxy node, the delegated R-node moves directly to replace the proxy node. Otherwise if failed sensor (event) is located at other places, all the nodes along the event and a selected R-node will have to be shifted towards another nearby sensor on path.

The author devises a localized Distance-Sensitive Node Discovery algorithm (D-SND), which mainly has two steps: information mesh construction and proxy node lookup. The mesh structure distributedly stores location information of all the proxy nodes and can be used for the purpose of proxy node lookup. Usually the iMesh is constructed under two types of models: well-structured grid network scenario or arbitrary network scenario. Then each proxy node distributes its own location information among the A-nodes along its residing row and column, and applies blocking rule for a pruned iMesh structure, illustrated in Figure 2.2. After constructing iMesh, proxy nodes should be discovered in order to find a nearest R-node to replace the failed A-node.

In this algorithm, the author relies on Greedy-Face-Greedy (GFG) routing protocol[5] for data forwarding. GFG is a guaranteed packet delivery protocol and is widely used in quorum-based location service. It searches by cross lookup, which means sending a search message in four directions along its residing grid row and column and comparing among its received location data to find a nearest R-node. Thus the selected R-node can be moved to compensate for coverage hole.



(a) A complete mesh structure (b) A (pruned) information mesh

Figure 2.2: An example of iMesh construction.

MSRP only solves the problem by replacing failed sensors with spare ones. It assumes full coverage of ROI before sensor failure, which indicates no polygon shaped sensing holes exist after initial sensor deployment. However, with random sensor node dropping, there is no guarantee that all sensors are well deployed with whole ROI covered by at least one sensor. Thus, performance of MSRP relocation is restricted.

2.2.3 Quorum-Based Approach

Wang et al. proposed a two-phase sensor relocation solution[32] with redundant sensors identified at first and then relocation to overcome the coverage holes created by failed sensors. The author locates the closest redundant sensor by a Grid-Quorum method. And then proposes a cascaded movement solution to relocate sensors in a timely and energy efficient way. Similar to MSRP[19], Quorum-based algorithm also focuses on fixing sensing holes created only by sensor node failure.

In order to find redundant sensors, the author divides the targeted ROI into grids, as shown in Figure 2.3. In the grid structure, the grids that need more sensors are the subscribers, while that have redundant sensors are the publishers. The delay

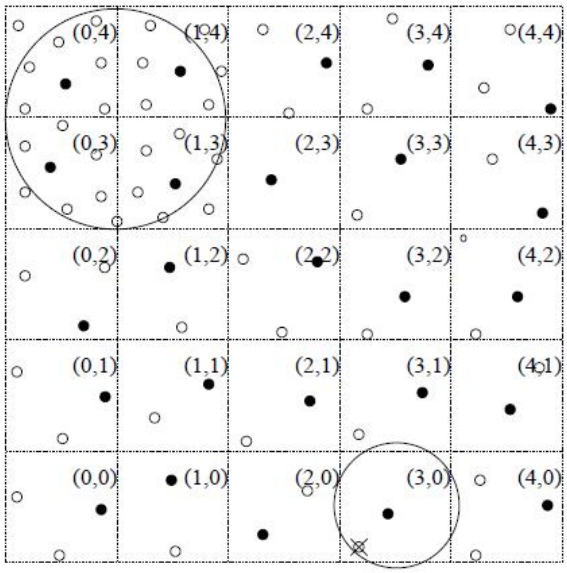


Figure 2.3: A quorum based system model.

may be relatively long if running on broadcast request approach. Therefore, it would balance between message complexity and response time if the matchmaking of subscribers and publishers happens at some intermediate grids. By moving one or more sensors, depending on response time, effect on current sensing topology, total energy consumption and minimal remaining energy, along with the redundant sensor, time and energy is saved for each single sensor. In Grid-Quorum method, the quorum belongs to the supply coterie intersects with all quorums in the demand coterie, and vice versa.

The first step of the proposed algorithm is to find a target redundant sensor. In order to construct a Grid-Quorum and find nearby redundant sensors, the author establishes supply quorum with those grid heads belonging to grids in one row, and demand quorum with those in one column. Since every demand quorum has intersection with all supply quorums, the grid head can get all the information about redundant sensors. To further reduce the message complexity, we add a stopping cri-

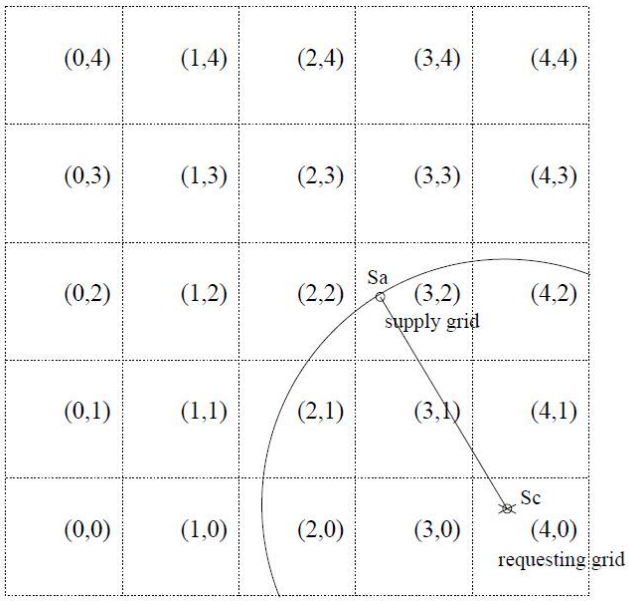


Figure 2.4: The stopping criterion.

terion for propagating request messages through the demand quorum, which means the propagation should stop once we get the best solution, see in Figure 2.4.

Once the redundant sensor is found in vicinity, the second step of the whole algorithm is about sensor relocation. The author proposes a cascaded movement in order to fit a preset maximum tolerate time as well as sensor energy saving purposes. This method is to find some cascading (intermediate) nodes and use them to help efficient relocation. The algorithm is based on a modified Dijkstra method, setting up two data structures, the primary search area and the waiting list, in order to reduce the message complexity for broadcasting. Each sensor is associated with a recovery delay, within which its successor must take its place in case other sensing coverage holes occurs while moving cascading nodes. Although the author's goal is to minimize the total energy consumption and maximize the minimum remaining energy, tradeoff should be considered in most cases. Normally the best way is the schedule with minimum difference between the total energy consumption and minimum remaining

power.

2.2.4 Restoring Internode Connectivity of Mobile Sensors

Younis et al. introduced a localized algorithm for Recovery through Inward Motion (RIM)[34]. In this algorithm, mobile sensors are used to recover failed sensors by moving towards detected location for the reported sensor. The authors aim to maintain WSN connectivity among all nodes after node failure. Again this algorithm concentrates on fixing sensing holes that are only caused by failed sensors in an already well deployed ROI.

The network may get partitioned if a sensor node failure happens. RIM restores the connectivity of the sensor network after the failure of a node by relocating the neighbors of the lost node. RIM decreases the travelling distance for individual sensors in case to balance energy consumption for different sensors. Otherwise one individual sensor may have a serious influence on the performance, causing reduction of the lifetime for this mobile sensor.

As a distributed algorithm, each sensor maintains a list of one-hop neighbours. When detecting failure of one neighboring sensor, cascaded node relocation method is used to move the neighboring sensors inward on the edge between the failure node and each neighboring node after sending message about location changes to its own neighbors as shown in Figure 2.5. In that way sensors will follow its neighbor inward until topology connectivity is repaired. Although connectivity is maintained, coverage hole may still stay uncovered if relation between sensing radius r_s and communication radius r_c is $r_s < r_c$. Besides, more sensors have to move in this algorithm, which may affect the stability of network.

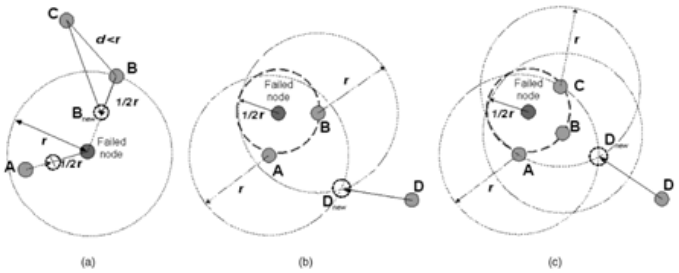


Figure 2.5: RIM relocation system in order to sustain connectivity to neighbours.

2.3 Coverage Repair in WSN by Robots

2.3.1 Traditional Approaches to Sensor Relocation

Mei et al. proposed three robot coordination protocols[25] for coverage repair: a centralized algorithm, a fixed algorithm and a dynamic algorithm. In the centralized algorithm, there is one robot acting as a central manager with rest of them as maintainers. Updated messages about latest positions are reported to central manager. Upon a failure report, the manager selects a robot based on current location of every robot and chooses the closest one to repair coverage hole with a functional nodes. In the fixed distributed protocol, the sensing field is divided into equal-size sub-regions, with the number of regions the same with the number of robots. A robot is in charge of a sub-region. This fixed distributed algorithm can avoid single manger bottleneck problem, but traveling distance is prolonged compared to centralized algorithm. In the dynamic distributed protocol, there are no fixed boundaries between robots. Voronoi diagram is constructed as a way to set dynamic boundaries. Similar to fixed distributed protocol, one robot is set in each Voronoi sub-region and nodes reports coverage holes to robots in their own Voronoi cells as shown in Figure.2.6. When a robot moves to repair the coverage hole, Voronoi graph needs to be adjusted dynamically.

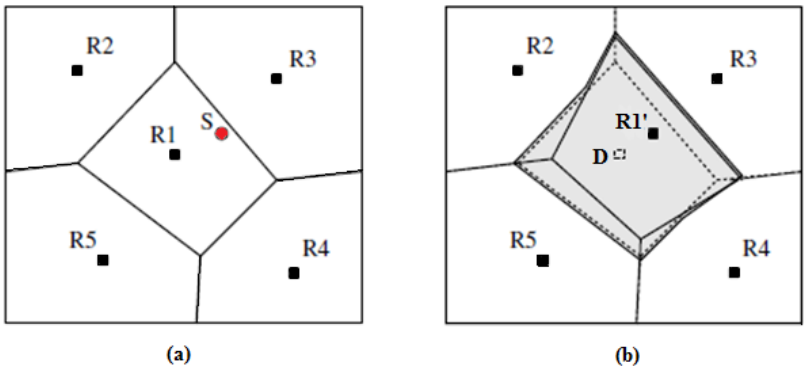


Figure 2.6: Voronoi diagram for dynamic region division in robot-assisted sensor relocation.

2.3.2 Firefly Swarm Optimization Based on Team Robots

Falcon ea al. proposed a centralized method for periodic replacement of damaged sensors by a team of mobile robots[10]. Due to the author's method, a small team of robotic agents lying at a base station work as vehicles to move passive sensors to replace damaged sensors. In the artificial firefly scheme, each robot follows the exploratory principles featured by Harmony Search. The goal of this method is to satisfy the demand of all delivery nodes by visiting as many pickup nodes as needed while minimizing the overall travel cost. A vertex, in this case referring to either delivery node or pickup node, can only welcome one vehicle visiting exactly once.

As the phenomenon of firefly swarm, a single firefly will always join towards any other individual brighter than itself. By applying firefly algorithm for sensor relocation, the author introduced the definition of fitness function and harmony search. The main idea of this algorithm can be envisioned as a firefly swarm in which each member follows the exploratory principles featured by harmony-seeking method. In order to reduce the time when browsing across the feasible solution space, the author conduct the search starting from very poor, infeasible solutions and gradually drive them into feasibility under the influence of a weak Pareto dominance relationship.

2.3.3 Reactive and Proactive Approaches

Magklara et al. introduced two local robot-assisted algorithms[24] about sensor relocation under static sensor network. The first algorithm is Reactive algorithm. In Reactive algorithm, the nodes send an alarm when their energy falls below a threshold. If more than one alarm are received by the robot, priorities are made by robot to decide which one to serve first. Weight is introduced in this algorithm to compare the priorities. In that case, high priority is given to the one that is closer to robot and has smallest remaining time until it fails. When there is no alarm occurring and robot not working, the robot remains idle and waits for the signal. Energy of all the nodes is updated in each round. Unlike Reactive algorithm, the Proactive algorithm does not use alarms and the robot never stops moving. Initially the robot moves randomly, discovering the active and passive nodes and checking their remaining energy at the same time. The main idea of Proactive algorithm is to try to avoid node failures, thus the robot keeps moving in sensing area. When the robot finds a passive sensor with enough remaining energy no far away from a sensing hole caused by node failure, it will swap these two sensors for coverage repair. Although sensing hole can be find with limited time delay, energy spent on robot for moving all the time may increase the cost for Proactive algorithm. Both Reactive and Proactive Approaches solve the coverage holes only caused by node failure.

2.3.4 Centralized Ant Colony Approach

Falcon et al. proposed a centralized optimization method: the one-commodity traveling salesman problem with selective pickup and delivery (1-TSP-SELPD)[9]. The main idea of this approach is that the demand of any delivery customer can be met by a relatively large number of pickup customers. In this case, not all pickup locations

have to be met in order to minimize to cost. Unlike 1-PDTSP[14], the topology is not a Hamiltonian cycle.

In most wireless sensor networks, there are some spare sensors acting as back-ups when some working sensors fail to work properly. Usually those spare sensors are called "passive" nodes which may go to "sleep" mode following a scheduling algorithm[13]. Thus, energy will be saved for these passive sensing nodes, since sensing task is one of the most energy consuming methods. As a centralized algorithm it is, a base station is set to collect and compare information from all the sensors. Periodically, every passive sensor reports its location to the base station and active sensors also report the coordinates of any adjacent sensing holes to the base station. Thus the base station can get the topology of all the delivery customers and pickup customers. Besides, base station is also the initial location of robot. In order to make the relocation process work in high performance, several rules should always be considered: 1). no nodes (other than base station) are visited more than once in a single tour; 2). all sensing holes must be repaired; 3). never violates the robot's capacity constraint. An example of finding an optimal relocation path for a single robot is shown in Figure 2.7.

Usually an ACO iteration comprises ant initialization, solution construction and pheromone update. Just as other ACO algorithms, pheromone value is updated at each iteration. Max-Min Ant System (MMAS)[30] is considered in this method to encourage the exploration of the entire space by placing a cap on the amount of pheromone on each edge in the whole network. Since dynamic pheromone bounds are used, every time a global best solution is encountered, the upper and lower bounds are changing according to given functions. Besides, heuristic function is computed to satisfy the constraints when selecting the next node considering the number of sensors already carried by robot. In this paper, several models for setting heuristic

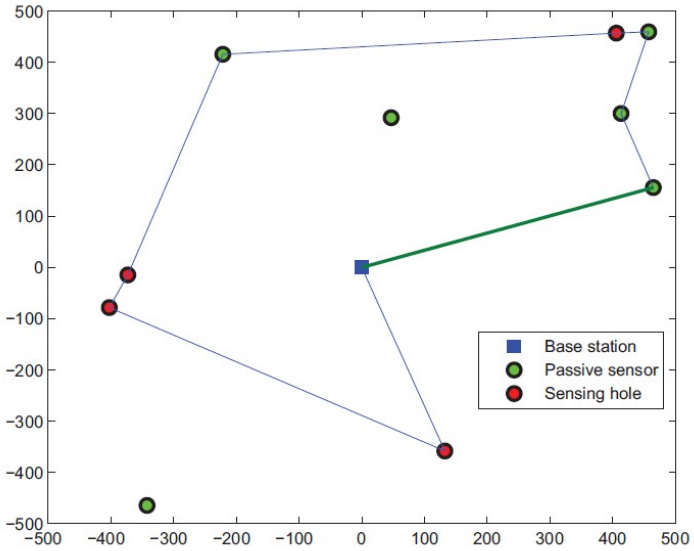


Figure 2.7: A feasible example for carrier-based coverage repair problem with Q_0=3 and Q=3. The green line represents departing direction.

values are introduced for further comparison. However, by applying 1-TSP-SELPD, only failed sensors are reported to be fixed by spare sensors. The algorithm simply ignores those bigger polygon-shaped holes after random sensor deployment.

2.3.5 Randomized Robot-assisted Approach

The existing algorithms introduced above deal with increasing coverage of WSN by monitoring ROI and replacing failed sensors to some passive back-up sensors. These algorithms assume initial well-deployment (full coverage) of sensors before detected node failure. However, through initial random sensor node dropping, there is no guarantee that all of the area within ROI can be monitored by at least one active sensor. Thus, the performance of above algorithms is limited to coverage maintenance after deterministic deployment. In order to fit in relocation of sensors after random deployment, polygon-shaped sensing holes should be considered and recovered in the method.

Fletcher et al. proposed a localized algorithm, Randomized Robot-assisted Relocation of Static Sensors (R3S2) and a grid-based variant (G-R3S2)[12] as an extension from [3]. Proposed algorithms focus on cover polygon sensing holes by relocating passive sensors within ROI.

Each sensor records the number of times that every adjacent grid point is visited by robots and recommends the least often visited grid to robot when a robot enters sensor's communication range. For each passive sensor, proxy node should be selected before falling "asleep". Whether the arcs on sensor's sensing range perimeter are covered or not by any active neighbour sensor can be identified through local computation, according to the location of neighbour sensors. Uncovered arcs indicate uncovered areas (sensing holes) near by. Meanwhile, shape of sensing holes can be detected through the list of uncovered arcs. By counting the number of uncovered arcs, number of sensing holes can be identified, even though inaccurate at some time. For every robot, there is a discovery phase of predefined length between any two successive movement steps. Robot remains static and transmits a beacon message to its nearby active sensors about its current location. Upon receiving this beacon message, adjacent sensing holes and passive sensors are reported from nearby active sensors or proxy nodes to robot. Through message reply among robot and sensors, robot is capable of obtaining local topological information about sensors and sensing holes. Then robot can make decisions about picking up some corresponding sensors and dropping at predetermined locations to cover sensing holes or part of a sensing hole. Every time a robot picks up a passive sensor, it sends a message to corresponding proxy to remove this passive sensor from its list. However, the method about finding sensing holes is inaccurate that may not lead to fully coverage repair. Different from R3S2 in which robot traverses by random movement, G-R3S2 is implemented on a virtual grid. In order to reduce randomness of robot movements, *Least Recently Vis-*

ited (LRV) policy[2] is applied to guide the robot toward to next step by neighboring sensors.

As an extension to [12], C-R3S2 and C-G-R3S2 were presented by Li et al.[16] as the variants of R3S2 and G-R3S2. In those latter two algorithms, clustering and virtual force techniques are adopted to merge local sensing holes. However with the use of cluster head sensors, all sensors in ROI must be able to consist to a *Connected Dominating Set* (CDS), which indicates all sensors should somehow be connected through multi-hops.

2.4 Ant-based Data Clustering Algorithms

Data clustering algorithms aim to cluster and classify datasets based on some pre-known physical properties, which is the opposite goal of sensor relocation methods. By clustering algorithms, similar items (data) are located near on an output grid graph, while items with different properties belong to different clusters farther from each other. Clustering algorithms are particularly suitable to perform exploratory data analysis and require much investigation to improve the performance. Our algorithm is developed from the ant-based data clustering algorithms [31] [6], which are inspired by the "algorithm" of how real ants clean their nests and organize dead bodies in their colonies. Ant-based agents are set into the grid for picking up and dropping data objects according to some clustering principles. The goal is to group similar items in their attributes in neighbour regions of a two-dimensional output grid.

Vizine et al. proposed an Adaptive Ant Clustering Algorithm, short for A^2CA[31], to improve the performance compared to Standard Ant Clustering Algorithm (SACA)[23]. By applying A^2CA, datasets are classified and clusters are formed with similar items grouped together. Ant-based agents pick up or drop data

objects based on corresponding probabilities. The probability of picking up an object increases with low density and similarity in neighbourhoods and decreases vice versa. Objects will be dropped if high density of similar objects are detected in its vicinity at a location. If a loaded ant decides to drop the object it is carrying, it looks for the first empty vicinity if the current position is already occupied. Cooling schedule, progressive vision field and pheromone releasing and detecting are applied as modification to converge into a more robust clustering solution. With cooling schedule, threshold constant of picking probability will decrease geometrically after a preset cycle, leading to a slight decline on picking up probability, so that agents will focus on moving objects with larger dissimilarities to neighbours in later stages.

Chen et al. introduced a novel ant clustering algorithm based on Cellular Automata[6]. An artificial Ants Sleeping Model (ASM) and an ant algorithm for cluster analysis (A^4C) are presented in this method, which can be regarded as a combination of Cellular Automata and Swarm Intelligence. From definition in the paper, each ant has two states: sleeping state and active state. Ants, representing data objects, tend to turn to sleeping state when surrounded by similar objects and active state when feeling "unconformable" about neighboring area. State of ant is controlled by a fitness function to its located environment and the probability of each ant turning active or staying asleep. The fitness function is defined to measure similarity of an ant with its neighbors and is determined by local information. When an ant has a low fitness function value, it has high probability to be active and search for a more secure and comfortable position to sleep. By ants self-moving to a more "comfortable" place to sleep, different subgroups are formed and data objects are clustered adaptively at end.

Chapter 3

Localized Ant-based Relocation Algorithms

As introduced from Chapter 2, most existing algorithms related to sensor relocation field focus on failed sensor recovery and replacement. This is a different relocation purpose compared with the proposed algorithms in this chapter, since we focus on relocating sensors after random sensor deployment and covering sensing holes caused by stochastic sensor nodes dropping.

R3S2 and its extended algorithms (G-R3S2, C-R3S2 and C-G-R3S2) [12][16] are the only existing solutions aiming at solving the same problems as the proposed algorithms in this chapter. As localized algorithms, robots obtain topological information through communication with neighbour sensors each time it locates at a new place. Spare sensors and sensing holes are locally detected by robots. Then robots pick up spare sensors and walk under some predetermined protocols to relocate them. Sensing coverage will be improved since sensing holes are at least partly repaired. However, the number of spare sensors in ROI are limited, especially when involved sensors are restricted. Thus, by only relocating spare sensors, performance of sensing coverage

increase is influenced if no more back-up sensor are added to the ROI.

In this chapter, we aim to design a *Localized Ant-based Sensor Relocation Algorithm with Greedy Walk* (LASR-G) which can increase sensor coverage in ROI after random deployment without adding extra sensors. Different from existing R3S2 algorithms, LASR-G relocates both spare sensors and some active sensors according to some principles to reduce overlapping of sensing range among active sensors within ROI. We then propose two algorithms based on LASR-G focusing on different concerns in relocation process. The first algorithm is LASR-G1, which is suitable if we only care about the coverage improvement, even if a robot travels longer distance to pick up a sensor. However if optimization of both coverage improvement and robot traveling distance is considered, a modified algorithm, named LASR-G2, is presented in the paper. The main difference between two proposed algorithms is how an unloaded robot chooses to pick up a targeted neighbour sensor.

3.1 Preliminaries

3.1.1 Inspiration

The proposed *Localized Ant-based Sensor Relocation Algorithm with Greedy Walk* (LASR-G) is inspired from *ant-based data clustering algorithms*[31][6]. The original goal for those data clustering algorithms is to classify datasets based on the properties. On the 2-dimensional output grid graph, the data objects with similar properties tend to locate nearer than those that are different in most aspects. Thus clusters are formed, with each consisted by similar data objects. Ant-based agents are applied in some clustering algorithm[31], carrying data objects which feel "uncomfortable" at original locations to search to join a better cluster. Agents picking up and dropping

decisions of data objects are based on corresponding probabilities. Fitness function is introduced to measure the similarities of each data object with its neighbour objects. With higher degree of dissimilarities among all properties of an object to its neighbour objects, the larger probability this object will be picked by agents. If loaded agent finds at a new location that neighbours are similar to the loaded object, it has high probability to drop the object at that location.

Similar to probability-based data clustering methods, LASR-G also achieves the relocation approach by probability-based robot decision making. Other than those clustering algorithms which aim to cluster similar data objects, LASR-G separates sensors and balances overall sensor density within ROI. Although achieving the opposite goal, our proposed LASR-G also applies modification schedules introduced in [31], which will be illustrated in detail in later section.

3.1.2 Network Model

We model our WSN by a 2-dimensional topological graph $\mathcal{G}=\{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges among each two $V_i, V_j \in \mathcal{V}$ if distance between V_i and V_j is within maximal communication radius. If $\exists E_{ij}$, which represent the edge between vertices V_i and V_j, then V_i and V_j are physically able to communicate directly and can be defined as one-hop neighbour of each other. $\forall V_i \in \mathcal{V}$, V_i measures its neighbour vertices set $\mathcal{V}'(i)$ as Eq. (3.1):

$$\mathcal{V}'(i) = \{V \in \mathcal{V} | V \neq V_i \bigcap (V, V_i) \in \mathcal{E}\}. \tag{3.1}$$

Here we refer vertices as sensors in a WSN, and we will rename sensor set as S for more accurate explanation in later section.

Since we do not consider robot path choosing with the effect of obstacles in our

proposed algorithms, we simply ignore the existence of obstacles and assume that robots are able to reach any location by walking a certain distance through any path as long as they are within ROI. Thus robot path choosing is only influenced by distribution of sensors in the network. For an unloaded robot (robot without carrying sensors at current time), it tends to walk to the area with high sensor density, while a robot has a higher probability to move where less sensors exist when the robot is loaded with a sensor (carrying a sensor at that time).

3.2 Assumptions and Definitions

Some definitions are made for the WSN along with assumptions for our proposed algorithms. Since our proposed algorithms focus on relocating targeted sensors after random sensors deployment, a set of sensors should be labeled for later relocation process. As localized algorithms, there is no base station or other devices for collecting all related information of WSN and sharing to robots and sensors. Thus, multi-robots scenario is preferred because with more robots acting as actuators in ROI, more areas will be monitored at same time, as well as more local information collected. Performance of single robot and multi robots will be compared in detail in later chapter to prove it. Thus, sensor set and robot set should be defined to identify each individual.

A wireless sensor network (WSN) is randomly deployed in the ROI, with its boundary information known to all nodes (including both robots and sensors). We define $S = \{s_0, s_1, ..., s_{n-1}\}$ as all the deployed n sensors. A fixed number of m robots (ants since it is inspired from ant colony based algorithms) $A = \{a_0, a_1, ..., a_{m-1}\}$ are applied in ROI, for replacing sensors. As robot-assisted algorithms, all sensors should be static ones with no locomotion attached to any of them. The move and replacement

of sensors solely relies on the help from robots. In each iteration, robot can pick up or drop a sensor. The maximum number of sensors a robot can carry, N, is same and limited for each robot. Here we assume $N = 1$, which indicates the maximal number of sensors a robot can carry at each time is 1. Thus once a robot picks up a sensor, it reaches full cargo and has to drop it before picking up another sensor. Based on current cargo, robot has two *load status*: "loaded with sensor" and "unloaded". Once robot picks up a sensor, its load status turns to "loaded" and has to drop loaded senor first, then it turns to "unloaded" again and searches for targeted sensor in a local way.

Normally battery for sensors and robots are limited. For each sensor, energy consumption happens most when sensor is executing sensing tasks if it is "awake" (active sensor). Thus for energy saving reasons, we set passive sensors to "sleep" and no longer monitor its local area. For each robot, it will have energy cost when it walks and calculates corresponding value for relocation purpose. The ones (both sensors and robots) that drain their battery fastest will be regarded as bottleneck of the whole WSN and turn failure once running totally out of battery. The failed nodes will no longer be able to fulfill any of the tasks and need to be recovered by some back-up nodes. The appearance of failed nodes will make the relocation work more complex. Since our LASR-G algorithms focus on the robot-assisted redeployment of sensors after initial random sensor node dropping, we can simply consider that dropped sensors are fully charged and still have enough remaining battery when fulfilling relocation tasks. As for robots, because of physical differences compared to sensors, they tend to have much larger energy storage. Thus, in this paper, we assume sensors and robots always have enough remaining energy to complete relocation process.

We assume physical properties (e.g. sensing radius, battery capacity, computational ability and message storage, etc) for all the sensors are same; all robots (for

multi robots scenarios) are also identical in the listed properties above. We define sensing radius for every sensor as r_s, sensor's communication radius r_c and robot's communication radius R_c. As the physical construction of most sensors and robots, we define the relation among r_s, r_c and R_c as Eq. (3.2):

$$r_s < r_c < R_c. \tag{3.2}$$

The relation indicates that for any 2 sensors s_i and s_j, the connectivity (able to communicate with each other) cannot simply guarantee their sensing perimeters intersect with each other. Definition of communication radius (r_c for sensors and R_c for robots) if a corresponding node is the farthest distance this node is able to reach and send messages to. For example, the farthest distance a sensor can send messages to at the points with the distance of r_c to sensor's current location. Similar, a robot is only able to cover the area shorter than R_c from its current location with messages, as indicated in Figure 1.5. In this case, for each robot, it can only receive messages from sensor whose distance is shorter than r_c to this robot.

For simplicity, we assume that sensing and communication ranges are a circle-shaped area, centered at sensor or robot, respectively. Area within the sensing range of an active sensor s_i is monitored by s_i. Here we assume is piece of subregion is well covered if it is fully monitored by at least one active sensor. In this case, the impacts in a subregion of single sensor monitor and multi sensor monitor are same in our assumption. A node (either a sensor or a robot) p_i inside communication range of another node p_j is capable to receive messages from p_j. Nodes communicate with each other by sending and replying "hello" messages. Each message contains the unique ID and location of the sender. If a message reply was received by a node, it stores sender's information and regards sender as one of its neighbour nodes.

By communicating with neighbour sensors, a sensor can decide whether its sensing range is fully covered by other active sensors, and consequently classify itself as a spare sensor. If so, it turns to "sleeping" mode to save energy. In "sleeping" mode, a sensor no longer monitors surrounding area within sensing range. However it still periodically keeps communicating with neighbour sensors in time synchronized manner. If it detects neighbour sensor failure or removal, it recalculates its sensing area, and turns active if no longer fully covered by other neighbour sensors.

3.3 Algorithm LASR-G1

3.3.1 Background

In a ROI, a WSN is constructed by stochastic sensor node dropping, part of ROI is shown in Figure 3.1. By random sensor deployment, there is no guarantee that all sensors will be well deployed with no sensing holes exist in ROI. In the opposite, sensing range of sensors will more or less overlap with each other, causing a waste of monitored area, meanwhile sensing holes may still exist where no sensors are in charged of. Take an example of Figure 3.1, most sensing area of sensor s_8 is overlaid with s_2, s_3 and s_4, while the area surrounded by sensing perimeters of sensor s_1 to s_8 is not monitored by any single sensor, creating sensing hole in the WSN. Since a robot a is located in ROI at d_0 in Figure 3.1, a is able to pick up one of these sensors according to some predefined rules and relocate to cover the sensing hole.

When initial random deployment is finished, robots travel within ROI to detect density and overlapping of sensors. Robots periodically flood "hello" messages to neighbour sensors at current location. When a robot finds a nearby sensor whose sensing range is mostly overlapped with other sensors, as the targeted sensor s_8 shown

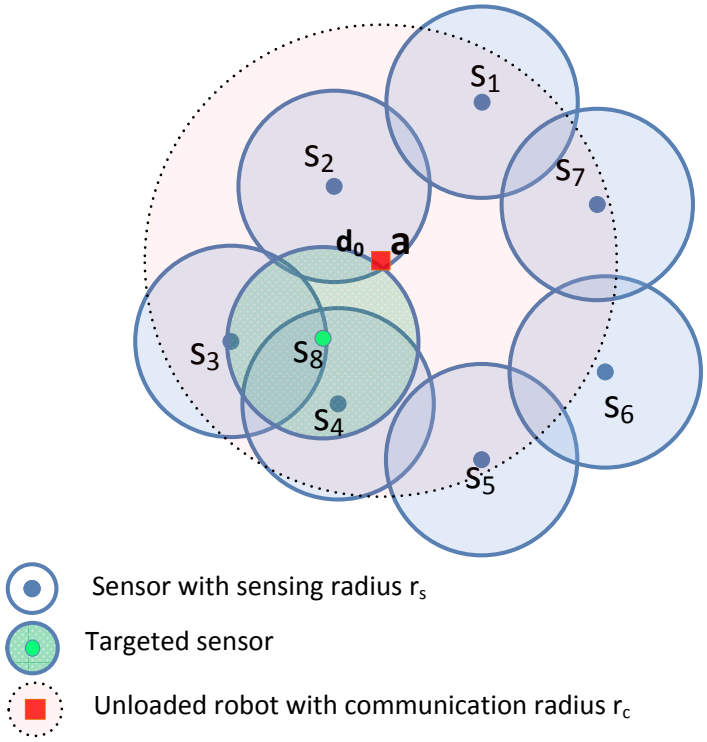


Figure 3.1: An example of robot and sensor locations before sensor relocation.

in Figure 3.2, it picks up this sensor and drops it where less overlay region appears. Robot picking decision is based on picking up probability calculated at current location. How robot decides to pick up s_8 from all neighbour sensors will be illustrated in detail later. The robot moving routes and coverage improvement is illustrated in Figure 3.2 and Figure 3.3. In Figure 3.2, robot a first walks directly to s_8 after calculation and selection of targeted sensor, then takes 3 consecutive steps towards destination. We define the step length as the distance robot can travel within each time window. We set step length the same for all step during the whole relocation process and represent step length as l, with $l < r_c$. Figure 3.3 shows the new covered area (shadowed area in Figure) after dropping off loaded sensor. Here we assume distance of targeted sensor movement is longer than r_s and usually takes several steps. Therefore we ignore the overlap of sensing range before and after moving and simply

consider the newly covered area after relocation as the increase of sensing coverage.

3.3.2 Robot Decision Making of Picking up and Dropping Sensors

For each robot, it makes picking up and dropping decisions based on its load status while walking within ROI. Both picking up and dropping decisions are solely depending on probabilities. As a localized algorithm, robot measures corresponding probabilities according to information within communication range.

Instead of involving traveling distance by robot, we describe first our solution LASR-G1 that only considers coverage improvement as the only optimization criterion. Let S' represent neighbour sensor set, that is, the set of sensors whose distance to robot a is less than sensor's communication radius r_c. $\forall s' \in S'$, robot a is able to receive messages from s' and thus can calculate *coverage ratio* (as indicated in Figure 3.3) of area centered at current location of s' (location denoted as d), shown in Eq.(3.3):

$$f(s', d) = \frac{Q(s', d)}{Qt}. \tag{3.3}$$

where $Q(s', d)$ is the area currently only monitored by sensor s' at its location d, which also equals to the possible newly created sensing hole area if s' is picked by robot a, as shadowed area in Figure 3.3. Qt is the total sensing area of s', which is a circle-shaped sensing range centred at current location of s' (by our homogeneity assumption, it is a same for all sensors). This *coverage ratio* can be calculated by dividing ROI into small grids by rows and columns, and measuring the corresponding area by counting the points within that area. The same equation can be also used to find the coverage ratio for newly covered area, if we relocate s' and place it at location d (in both cases, we assume that previous and new coverage areas for s' do

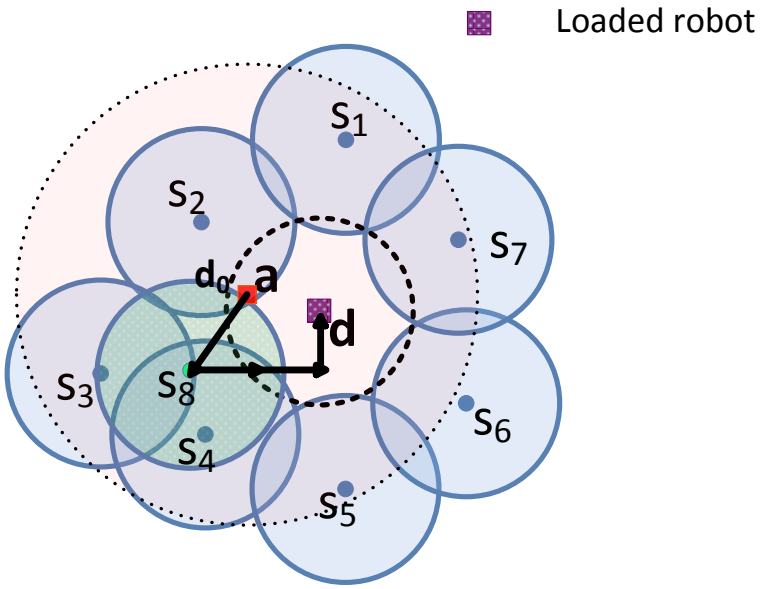


Figure 3.2: An example of how robot a picks up and drops a sensor. Dashed circle is the possible constructed sensing range if loaded sensor is dropped at current location. Three consecutive steps are taken based on predefined path selection method to find a comfortable place to change load status.

not overlap).

By calculating and comparing *coverage ratio* for every sensor $s' \in S'$, robot a is able to find a neighbour sensor st with minimal *coverage ratio* value, since the one with minimal value has least contribution to coverage within ROI. If more than one sensor ties on *coverage ratio* value, robot simply chooses the nearest neighbour sensor as st.

Based on *coverage ratio* value in Eq.(3.3), the probabilities of robot a picking up and dropping decisions are given as Eqs.(3.4) and (3.5), respectively:

$$Pp(a,d) = f(st,dt)^{ep}, \tag{3.4}$$

$$Pd(a,d) = f(sl,d)^{ed}, \tag{3.5}$$

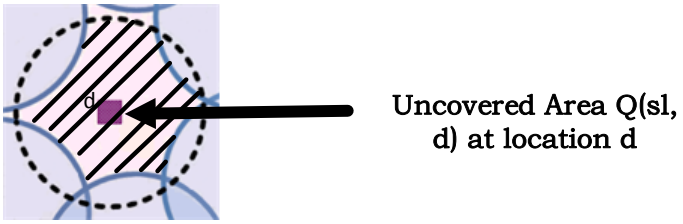


Figure 3.3: Description of *coverage ratio* and newly created sensing holes. By dividing the area into grids through rows and columns, area can be measured by counting number of points within measured region.

where d is the current location of robot a, st is the targeted neighbour sensor with minimal *coverage ratio*, dt is the location of st, sl is the currently loaded sensor of a. Here we rename the loaded sensor as sl, since it is not guaranteed that targeted sensor st will be picked and loaded all time. If picking probability is low, it is possible robot finally decides to leave st in topology without replacing to anywhere else. $\frac{Q(sl,d)}{Qt}$ is the newly added *coverage ratio* if sl is dropped at d. As shown in Figure.3.2, when a loaded robot a arrives at a new location d, it calculates dropping probability before taking any action. If robot decides to drop sensor sl, it leaves sl directly at its current location. Centred at d, $Q(sl, d)$ equals to the current uncovered area within r_s, while Qt is the total area within dashed circle.

Parameters ep and ed control picking up and dropping probabilities respectively. Both ep and ed follow a modification schedule: for each cycle (e.g. 100 steps), the value of ep increases and ed decreases by a geometric progression scheme, controlled by constants vp and vd shown in Eqs.(3.6) and (3.7), until it meets the preset stopping criterion.

$$ep \leftarrow ep \times vp, \tag{3.6}$$

$$ed \leftarrow ed \times vd. \tag{3.7}$$

where vp and vd are two preset constants that control the modification schedule of

parameters ep and ed.

The modification schedule favors an adaptive relocation process. Robots focus on recovering larger sensing holes at the beginning of the process, even though it may create smaller new holes by moving targeted sensors. After several cycles of relocation implemented, most large connected uncovered areas tend to be monitored by sensors, partly covered and then separate into several smaller holes. Then we need to reduce probability for newly created sensing holes by increasing ep, and enlarge dropping probability by decreasing ed to cover small holes. In order to increase ep and decrease ed at same time, constraints are made for constants vp and vd: $vp > 1$, $vd < 1$.

Each time robot a calculates a corresponding probability P, it compares P with a random number r from a uniform distribution in [0,1]. If $r \leq P$, robot a decides to either pick up or drop a sensor, according to its current load status. Otherwise it simply takes a step to another nearby location in ROI following a step selection process, to be described later.

3.3.3 Robot Path Selection

As a localized algorithm, only local information is obtained by each robot. In this case, there is no central control (as base station in most existing centralized algorithms) to guide robot to global optimal path. Thus, robot is not able to come up with a global best traveling route for sensor relocation. However, total random route within ROI will increase the time and energy consumption for robot to both find targeted sensors and drop loaded sensors at "comfortable" places. Therefore, we propose a local greedy path selection protocol to reduce randomness in robot traveling route.

LASR-G improves the performance by adding some local greedy walk to reduce

total randomness on robot movements, decrease total traveling distance and shorten time delay for relocation process. After a picking up or dropping action is finished, robot changes load status and takes a greedy step to a more "comfortable" place (usually one of the locations in its vicinity, will be illustrated in detail later) with a preset step length l to repeat the tasks. However, if a picking up or dropping task is not successful, robot keeps the same load status and walks to search for other suitable places, to which both greedy walk and random walk may be taken, with random walk happens when a preset continuous number of greedy steps reached without successfully changing load status.

With or without carrying sensor, robots needs to decide where to move to. The set of neighbouring sensors $|S'|$ dynamically changes. Greedy movements focus on the selection among four directions: north, south, east and west. Based on local information, robot a estimates fitness function values at four points dn, ds, de and dw, respectively, each at distance l from current location d. Fitness function is either picking up probability $Pp(a, dx)$ or droping off probability $Pd(a, dx)$, where dx is one of four candidate locations de, dw, dn, and ds. As shown in Figure 3.4, robot at current location d has information of sensor s_1 to s_5. It is not able to accurately measure fitness function at de because it is not aware of neighboring sensor s_6 if robot locating at de. However robot a estimates that s_1 is not a neighbor of de, and calculates fitness function based on estimated neighborhood s_2 to s_5 (sensors in the intersection of two sensor communication ranges, drawn in darker color) of de. Robot a compares fitness values at four selected candidate points, and chooses one to move to according to current load status. After moving to the new location, robot repeats *coverage ratio* and corresponding probability calculation and move selection.

An example is given in Figure 3.2 for an overall introduction of LASR-G. Assuming robot a is unloaded at location d_0, coverage ratio for every neighbour sensor is

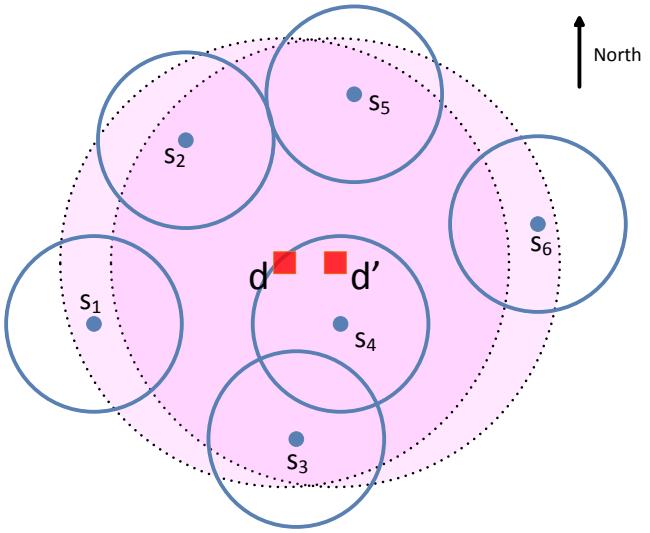


Figure 3.4: An example of fitness function estimate and greedy direction selection.

calculated and targeted picking up sensor st is selected. Then robot calculates picking up probability $Pp(a, d_0)$ based on neighbour sensors from s_1 to s_8. If picking decision is made, a chooses s_8 as targeted sensor and walks straight towards s_8 to pick it up. Then a chooses a local best step to walk away. Here it compares estimated fitness function value for four directions and chooses east-most direction for the first step. After calculating dropping probability at current location, robot decides to continue taking greedy walk with sensor loaded. When it arrives at location d, robot decides to leave loaded sensor at d based on corresponding dropping probability at d. Then unloaded robot a will switch to consider candidate pickup probabilities.

By simply applying greedy method, it may cause local loops and rotation in robot movements if robot keeps the same load status for several steps. This will lead robot to stuck in a small part of subregion other than wandering farther across ROI and finding an appropriate place to change load status. To expand vision, robot will switch to random move with a larger step length $q \times l$ (or q consecutive steps if consider robot taking uniform motion) if and only if a preset number of steps t is

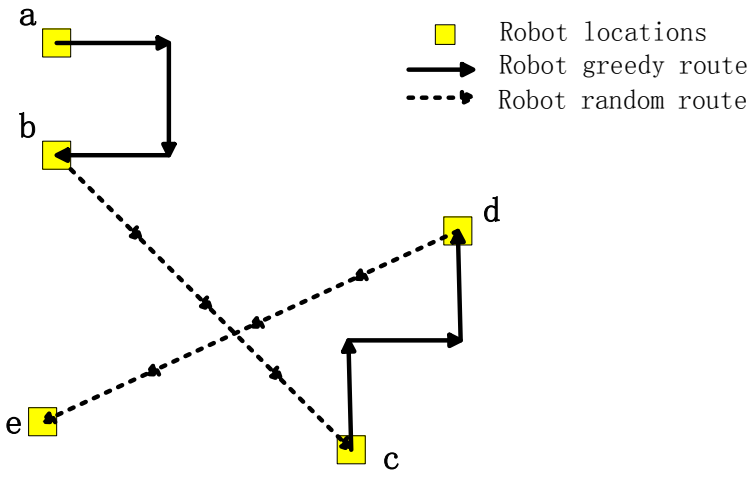


Figure 3.5: An example of robot movement without change of load status. From a to b, $t = 3$ greedy walk steps are implemented. Then robot changes to $q = 4$ steps of random walk from b to c in same load status. A repeated movement occurs from c to e. Robot keeps switching between greedy walk and random walk until the change of load status.

reached without any change of robot's load status. During random walk, robot will move in a straight path to a random direction without stop. It will switch back to greedy walk after the random move. Robot keeps switching between greedy walk and random walk until it finds a "comfortable" place to change load status. Then robot starts from greedy walk again. An example is shown in Figure.3.5. Starting at location a, robot takes 3 greedy steps to location b. Without changing of load status, robot switches to random walk until it reaches c. Then it switches between greedy walk and random walk until it reaches e where robot is able to change load status according to some calculation.

3.4 Algorithm LASR-G2

3.4.1 Background

For introduction above, it is confirmed that LASR-G1 only considers coverage improvement. However, as an improvement for energy saving purpose, both coverage and robot traveling distance are optimized in LASR-G2. Inspired by ant-clustering algorithm, pheromone is considered in LASR-G2, which is the fitness function value calculated by robot taking each step at each location. With higher fitness function value, robot has higher probability to pick up a targeted sensor chosen from neighbour sensors according to some selecting principles.

3.4.2 Robot Decision Making of Picking up and Dropping Sensors

At each step, robot collects information of all neighbour sensors during a time window. Then it can get a unique fitness function value at current location based on location of neighbour sensors and spare sensors. $\forall\ s \in S'$, where S' is neighbour sensor set at location d except sensor sl robot a is carrying if it is loaded, contribution on robot decision is calculated. Fitness function of robot a at location d is given below as Eq.(3.8),

$$g(a,d) = C \cdot \sqrt{ln|S'| \cdot e^{|S''|} \cdot \sum_{s \in S'} e^{\frac{-|d-D(s)|^2}{2\sigma^2}}}. \tag{3.8}$$

where C is a constant to be optimized, $|S'|$ is the number of sensors in set S'. S'' represents the set of spare sensors within communication range r_c, $S'' \subset S'$. $|S''|$ is the number of elements in set S''. $D(s)$ is the 2D coordinates of sensor s, $|d - D(s)|$ represents the distance between robot a and sensor s, σ^2 stands for the variance of

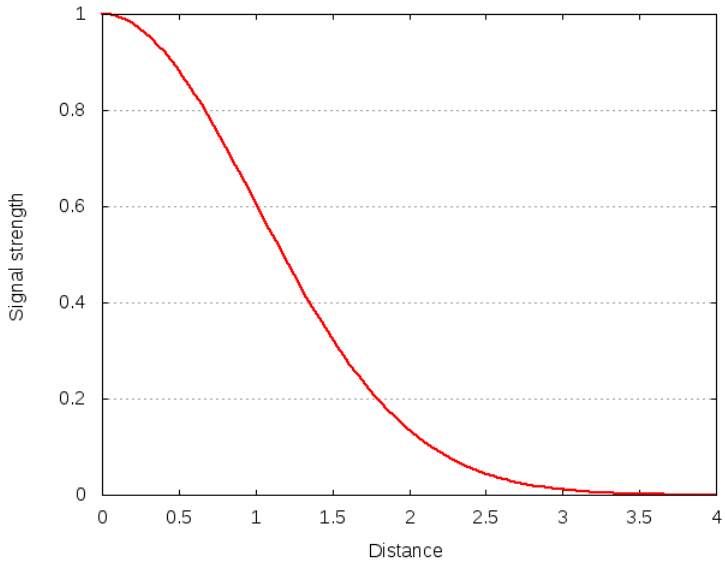


Figure 3.6: Gaussian activation function.

distribution of sensors to the robot. Influence of sensors outside communication range on fitness function can be ignored.

Here we use Gaussian activation function to express the relation between distribution of neighbour sensors and impact of robot picking up decisions. The trend of Gaussian function is displayed in Figure 3.6. $\forall s \in S'$, the contribution of s on fitness function is described as Eq.(3.9).

$$h(s,d) = e^{\frac{-|d-D(s)|^2}{2\sigma^2}}. \tag{3.9}$$

For all neighbour sensors, the ones closer to robot (the centre of communication range) have more impact on robot picking decisions than those near communication range perimeter. Robot takes shorter traveling distance if determining to pick up sensors nearer to robot itself. An example is shown in Figure 3.7 to illustrate the point. s_1 to s_4 are neighbour sensors closer to robot a compared to s_5 to s_8 which locate near current sensing perimeter. Since a takes less move reaching s_1 to s_4 than s_5 to s_8, if new created sensing holes are similar, or even those nearer have smaller newly created

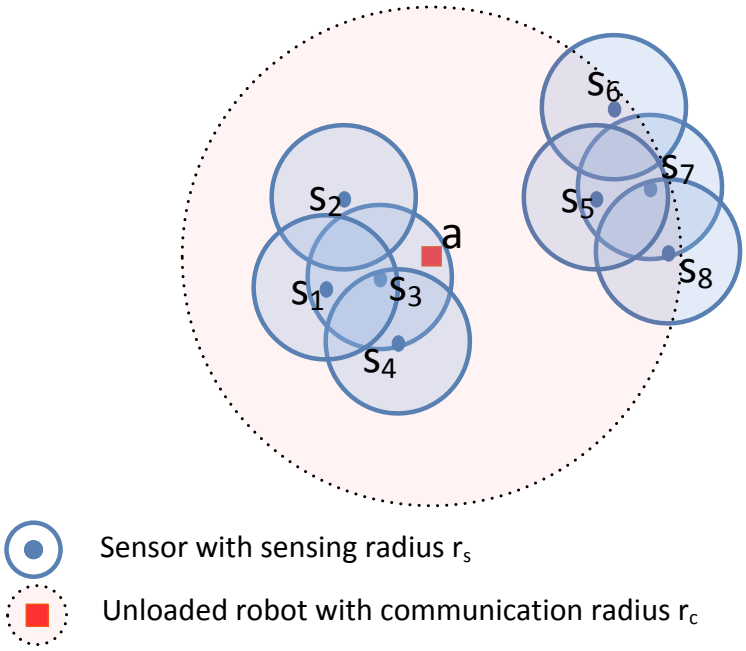


Figure 3.7: Impact of sensor distribution on fitness function value.

sensing holes, s_1 to s_4 have larger impact on robot picking up decisions comparing to the rest ones near perimeter.

Based on fitness function value in Eq.(3.8), the probabilities of robot a picking up and dropping a sensor at d are given as Eqs.(3.10) and (3.11), respectively.

$$Pp(a,d) = (\frac{g(a,d)}{1+g(a,d)})^{ep}, \tag{3.10}$$

$$Pd(a,d) = (\frac{Q(sl,d)}{Qt})^{ed}. \tag{3.11}$$

Dropping probability calculations for LASR-G1 and LASR-G2 are same. The main difference between the two algorithms is on how robot derives picking up probability and makes picking decisions. Unlike LASR-G1 which simply choose to pick up a targeted sensor with minimal *coverage ratio*, picking up probability in LASR-G2 is obtained by balancing coverage improvement and robot traveling distance. Similar

to LASR-G1, parameters ep and ed in LASR-G2 also follow a modification schedule referred in Eq. (3.6) and (3.7). Robot makes decisions by comparing corresponding probability to a random number r which subjects to uniform distribution in [0,1]. Let P represent corresponding probability (picking up probability or dropping probability depending on load status), if $r < P$, then robot chooses to take a task and change load status thereafter. If robot decides not to change load status, other than replacing any deployed sensor, it simply takes a step to another nearby location in ROI following a predetermined step selection basis mentioned later.

If robot a determines to pick up a sensor, it selects to pick up a targeted sensor following the given principle: Robot detects neighbour spare sensors and stores them each time it walks to a new place. Let V represent spare sensor set containing the ones that remain to be moved. Robot updates set V when neighbour sensors changed or more spare sensors detected. Since we focus on both improving coverage and minimizing robot traveling distance, Eq.(3.12) is applied for both factors.

$$Z(s', d, d') = (1 - \frac{Q(s', d')}{Qt}) \cdot e^{\frac{-|d-d'|^2}{2\delta^2}} \tag{3.12}$$

where d is current location of the unloaded robot a, d' is location of sensor s'. δ^2 is the variance of distance between a and s' on the impact of robot selecting decision. $\forall s' \in S' \cup V$, a unique value $Z(s', d, d')$ is obtained. The robot chooses the sensor which has maximal value as targeted sensor st. If more than one sensors tie on maximal value of $Z(s', d, d')$, robot chooses to pick the one nearest to robot's current location d. If selected st is an element from set V, it is then removed from V when picked by robot.

3.4.3 Robot Path Selection

Based on local information, robot a estimates fitness function values for the four directions (north, south, east and west) a step length l farther from robot a's current location d. As shown in Figure.3.4, robot is at current location d with information of sensor s_1 to s_5. If at next possible location d' a step length l east to current location, it is supposed to calculate fitness function based on sensor s_2 to s_6. As indicated before, since sensors near communication perimeter has limited contribution to value of fitness function based on Eq.(3.9), and since robot at d cannot get information outside its current communication range, it will simply estimate fitness function at $g(a, de)$ by information in the overlapping area of communication ranges for current d and east most location d' (dark pink area in Figure.3.4). Here de indicates the east candidate location a step length l away from current location, also indicating d' in Figure 3.4. Through estimation for greedy direction, when applying Eq.(3.8) for $g(a, de)$, S' and S'' are only estimated by information from overlapping region. After estimating four directions $g(a, dn)$, $g(a, ds)$, $g(a, de)$ and $g(a, dw)$, robot will compare the values and move to a local best place then. If loaded, it chooses the location with minimal fitness function value; otherwise it will move where with maximal value. After moving to a local best location, robot starts to communicate with neighbour sensors and repeats calculation and tasks again.

Chapter 4

Experimental Validation

We have conducted experiments to test the feasibility of our *Localized Ant-based Sensor Relocation Algorithms with Greedy Walk* (LASR-G1 and LASR-G2), and compare the proposed algorithm with *Grid-based Randomized Robot-assisted Relocation of Static Sensors* (G-R3S2)[16]. We analyse the performance for both algorithms based on four metrics. TD metrics measures traveling distance, while CP, RCP and MCP measure coverage percentage (in three different ways).

4.1 Performance Metrics

1. Traveling Distance (TD): For single robot, it refers to the total traveling distance until robot movement terminates. For multi robots, it indicates average traveling distance of all robots.

2. Coverage Percentage (CP): The percentage of area which is monitored by at least one sensor over the given ROI, when robot movement terminates. CP is measured when traveling distance (TD) for LASR-1 and LASR-2 reach TD value of G-R3S2 when it stops (achieving the best coverage G-R3S2 can get).

CP is used to for compare the coverage among algorithms when each robot walks same distance.

3. Real-time Coverage Percentage (RCP): The real-time percentage of area which is monitored by at least one sensor over given ROI. It is defined and measured when robot TD reaches some preset value. RCP is used to illustrate the impact of real-time coverage improvement under the impact of robot TD.

4. Maximal Coverage Percentage (MCP): The maximal percentage of area which is monitored by at least one sensor over given ROI. MCP is measured when increase of coverage value is less than 0.005 within 500 steps. MCP shows the maximal coverage percentage each algorithm can reach regardless of TD value for each robot.

4.2 Simulation Initialization

We generate the simulation of our work in Java and execute on an AMD Athlon 7450 Dual-Core Processor at 2.40GHz with 2.75GB of RAM under Windows XP. Sensors are initially randomly deployed within a rectangular 600×600 ROI, as shown in Figure 4.1. The number of sensors n and robots m are preset by users. Here we set $r_s = 25$, $r_c = 50$ and $R_c = 100$. All of the radiuses are measured by meters. Robot step length l is set to 10. Parameter values for this experiment are set as below: C=0.6, q=8, t=5. The initial value of ep, ed, vp and vd are set as 1.50, 1.15, 1.005 and 0.99, respectively, with values recurse shown in Eqs. (3.6) and (3.7) for every 200 steps. How to set σ and δ value will be discussed through experimental result. In simulation, we terminate G-R3S2 algorithm when no spare sensor exists or its coverage reaches 0.99. We test each scenario for 20 times and get average value

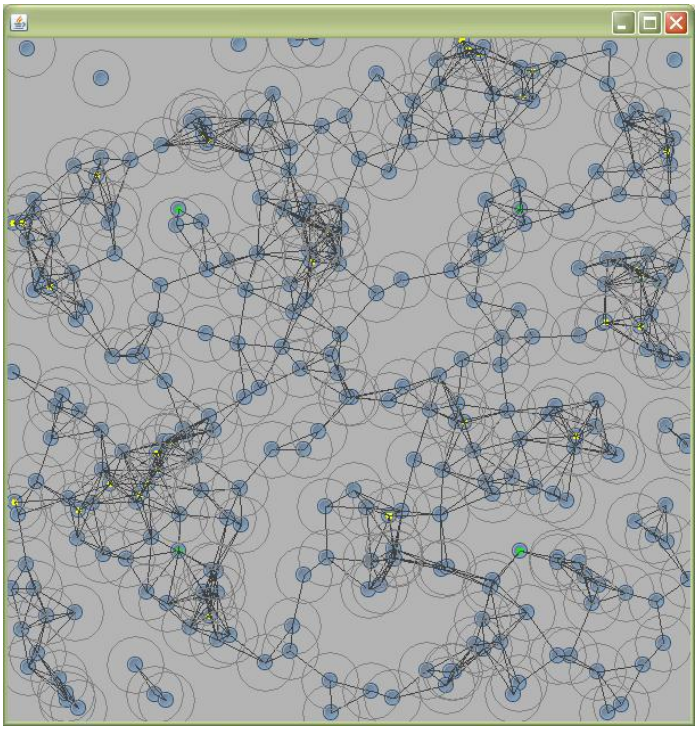


Figure 4.1: Sensor initial random deployment within a rectangular 600×600 ROI.

for corresponding tested metrics.

To ease the analysis, we do not consider run-time sensor node failure during relocation process in our simulation. We will compare the performance by modifying the number of sensors n and robots m, and compare coverage improvement and average traveling distance for each robot during different relocation periods.

4.3 Simulation Analysis

To validate our proposed algorithms (LASR-G1 and LASR-G2) and compare with the only existing algorithm G-R3S2 [16] that relocate static sensors after random initial deployment, we analyze all these algorithms according to the following aspects. Simulation results and applicable figures and charts are given in detail.

4.3.1 Parameter Value Selection

Some value of parameters are undecided for comparison with existing algorithm, e.g. σ and δ. We will simulate our proposed algorithms under different values of σ and δ, and select a most appropriate value for later use. σ and δ is only applied in LASR-G2 which affect value of picking probability and targeted picking up sensor selection. Thus we will only compare the RCP value for LASR-G2 under different σ and δ.

4.3.1.1 Impact of σ on LASR-G2 Coverage Improvement

Here we set δ=8, and compare RCP and CP performance for different number of robots m and sensors n when adjusting σ value. Possible figures will be displayed with explanations.

We first evaluate RCP performance when different traveling distance TD is taken by each robot. Here we only analyze the scenario with m=4 robots and n=350 sensors, as shown in Figure 4.2. RCP value increases with longer TD value for all σ. σ is defined in fitness function Eq.(3.8) as a variance of robot evaluation of surrounding neighbour sensor environment. As subject to Gaussian distribution, σ determines how neighbour sensors will affect robot fitness in vicinity according to distance between each sensor to robot itself. From Figure 4.2, it is confirmed that when taking same TD, RCP is highest if σ is set to 12 in this scenario.

Then we compare performance of CP when setting m and n to different values. For the convenience of later comparison to competing algorithm G-R3S2, we set the stopping criterion here for LASR-G2 when it reaches same TD as G-R3S2. Stopping criterion for G-R3S2 is when spare sensors exist in ROI or RCP reaches 0.99. The result is shown in Figure 4.3.

Figure 4.3(a) shows the relation of m to CP under different σ value. Here we set

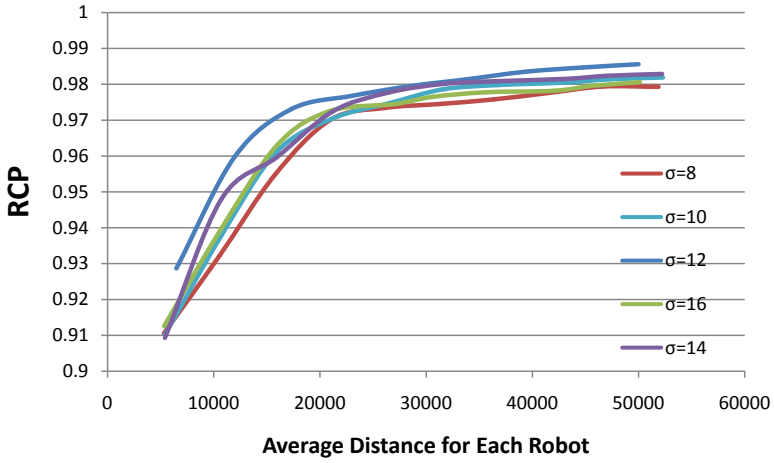


Figure 4.2: Relation between RCP and TD under different σ values.

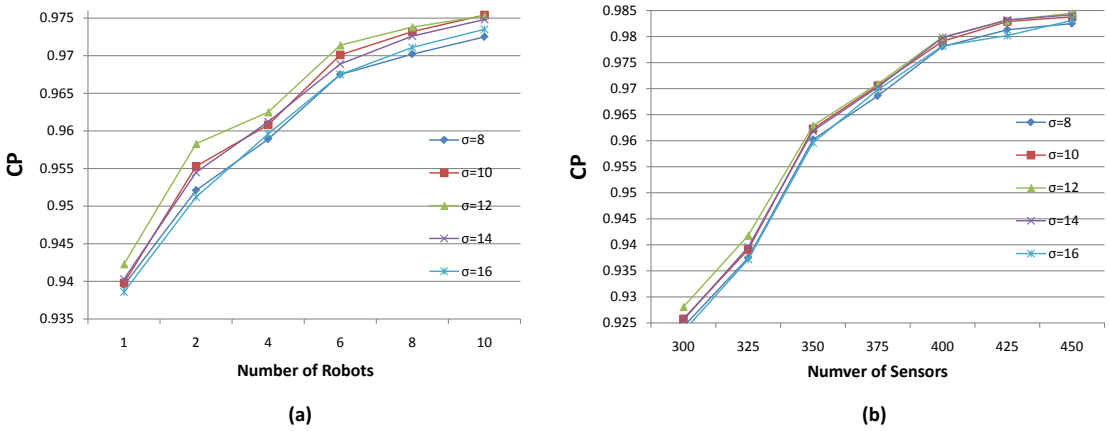


Figure 4.3: Impact of σ on CP value. (a) is the comparison for different m; (b) shown relation when applying different n.

n=350. It gives highest CP value when σ is set to 12, compared with situations when σ is other values. For Figure 4.3(b), we CP value is almost similar especially when more sensors are deployed in ROI. For the scenarios where less sensors involved, CP value is slightly higher when σ=12. Thus, we set σ=12 for LASR-G2 in later simulation evaluation and comparison with competing G-R3S2.

4.3.1.2 Impact of δ on LASR-G2 Coverage Improvement

From earlier analysis, here we set σ to 12. Then we evaluate impact of δ on RCP and CP value for different robot and sensor scenarios. Similar to σ, δ also is a variance in Gaussian distribution. It is used to balance newly created coverage holes and robot traveling distance to some specific sensor when robot has decided to pick up a sensor. With different δ values, robot tends to pick up different candidate sensors according to the Eq. (3.12).

Similar to methods of evaluating σ, we first analyze the relation between TD and RCP under different δ value. Number of robots m=4, number of sensors n=350. Simulation result is demonstrated in Figure 4.4. We can conclude from Figure 4.4 that LASR-G2 algorithm gain best RCP value when taking same robot TD in situation δ=8, if m and n are set as 4 and 350 respectively.

Then we analyze CP value when applying different number robot or sensors in ROI. Still the stopping criterion for LASR-G2 is when robot TD reaches the distance G-R3S2 stops. For Figure 4.5(a), scenarios of δ=8 outperform other δ value when using different number of robots for relocation. Similarly, when number of sensors n changes as indicated, it will reach maximal CP value, especially with less sensors involved, when setting δ as 8. If more sensors deployed in ROI, robot selecting targeted picking sensors will not be highly relevant to distance between current locations of robot and each candidate sensor, due to increasing picking options available for

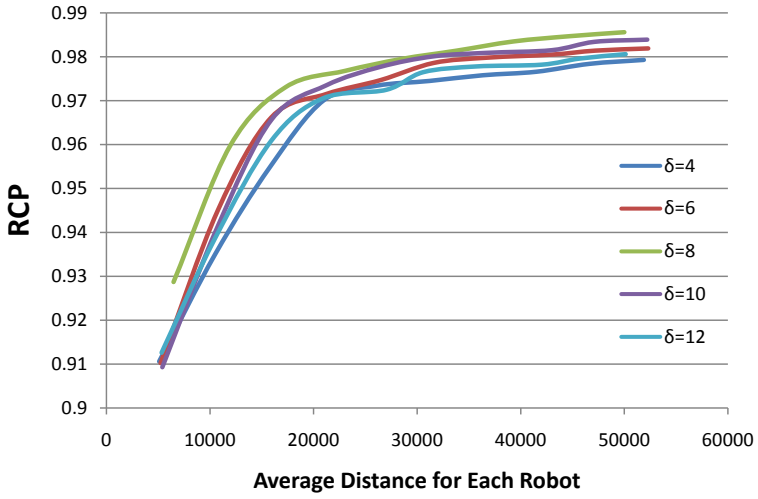


Figure 4.4: Relation between RCP and TD under different δ values.

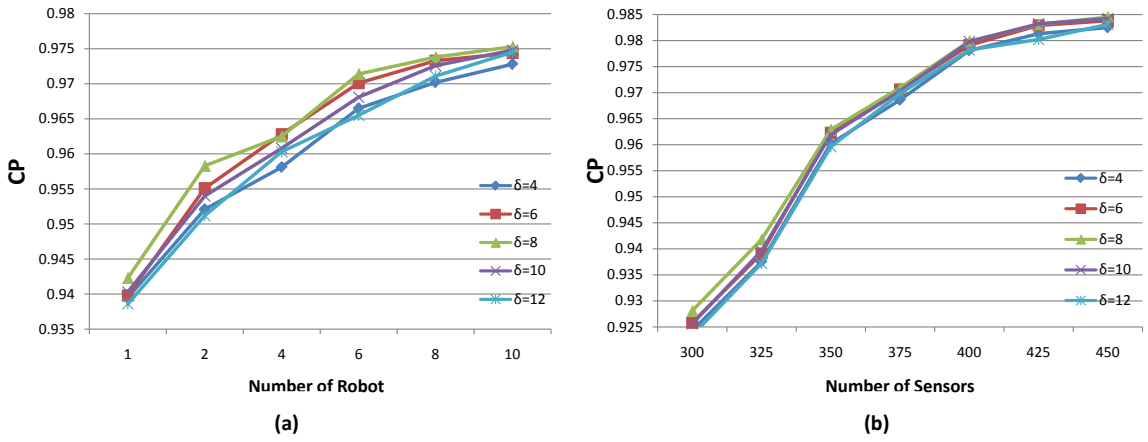


Figure 4.5: Impact of δ on CP value. (a) is the comparison for different m; (b) shown relation when applying different n.

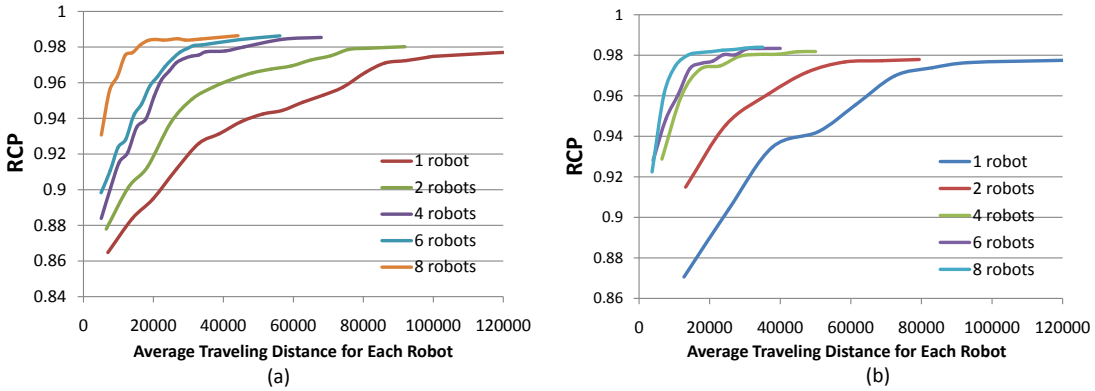


Figure 4.6: Single robot and multi robots scenarios: (a).LASR-G1; (b).LASR-G2.

robot. In that case, we set δ=8 for later simulation evaluation for LASR-G2.

4.3.2 Single Robot and Multi Robots Scenarios on Relation between RCP and TD

As number of robots preset by users, we compare the process of RCP increase and TD performances of single robot and multi-robots when applying LASR-G1 and LASR-G2. Within the tested ROI, 350 sensors are initially randomly deployed with preset number of robots m for relocation. Figure.4.6 (a) and (b) illustrate the relation between RCP and TD for LASR-G1 and LASR-G2 respectively when applying m robots in ROI. The maximal RCP value relocation process can get is the MCP (maximal coverage percentage) in given ROI for preset n=350 sensors and m robots.

For both algorithms, multi-robot relocation can construct better coverage improvement, since cooperation of team robots tends to reach a satisfactory coverage improvement for less average traveling distance (TD) than single robot scenario. With more robots involved in ROI, more rapid RCP increment appears for the WSN. Robot vision is extended by executing multi-robot over a ROI.

Comparing (a) and (b), LASR-G1 takes longer robot TD to reach MCP, with

final MCP value slightly higher than LASR-G2 when using same m robots. For both algorithms, MCP value has only slight increment when m increases, and average TD value for each robot decreases with the increase of m when reaching MCP, leading to less TD for each robot. However, with more robots encountered, it will increase the cost for the WSN. Besides, although average TD reduced, total TD value for all robots may still increase with the use of more robots. Thus, value of m is chosen depending on the bottleneck of the process. If robot average TD is the bottleneck, more robot should be involved to the process. While appropriate number of robots should be considered if the main bottleneck is cost and total energy consumption.

4.3.3 Impact of Number of Robots

We will compare LASR-G1, LASR-G2 and G-R3S2 algorithms over the measured aspects. 350 sensors are initially randomly deployed in ROI before relocation starts, with preset number of robots m for relocation. Figures 4.7(a), (b) and (c) illustrate the influence of m (number of robots) on CP, MCP and TD values for LASR-G1, LASR-G2 and G-R3S2. To simplify the simulation, we stop G-R3S2 when no more spare sensor exists within ROI. Stopping criteria for LASR-G1 and LASR-G2 are set differently for Figure 4.7(a), (b) and (c).

Impact of number of robots m on CP is presented in Figure 4.7(a). We evaluate CP under same average robot traveling distance. Stopping criteria for both LASR-G1 and LASR-G2 are set when TD value reaches the one measured when G-R3S2 stops. With the increment of m, LASR-G2 performs a rapid increase on final CP, while value for LASR-G1 and G-R3S2 show slower increment. Since we assume robot takes uniform motion during the whole relocation process, with shorter average robot TD, shoter recovery time is needed if we ignore the run-time sensor node failure. This

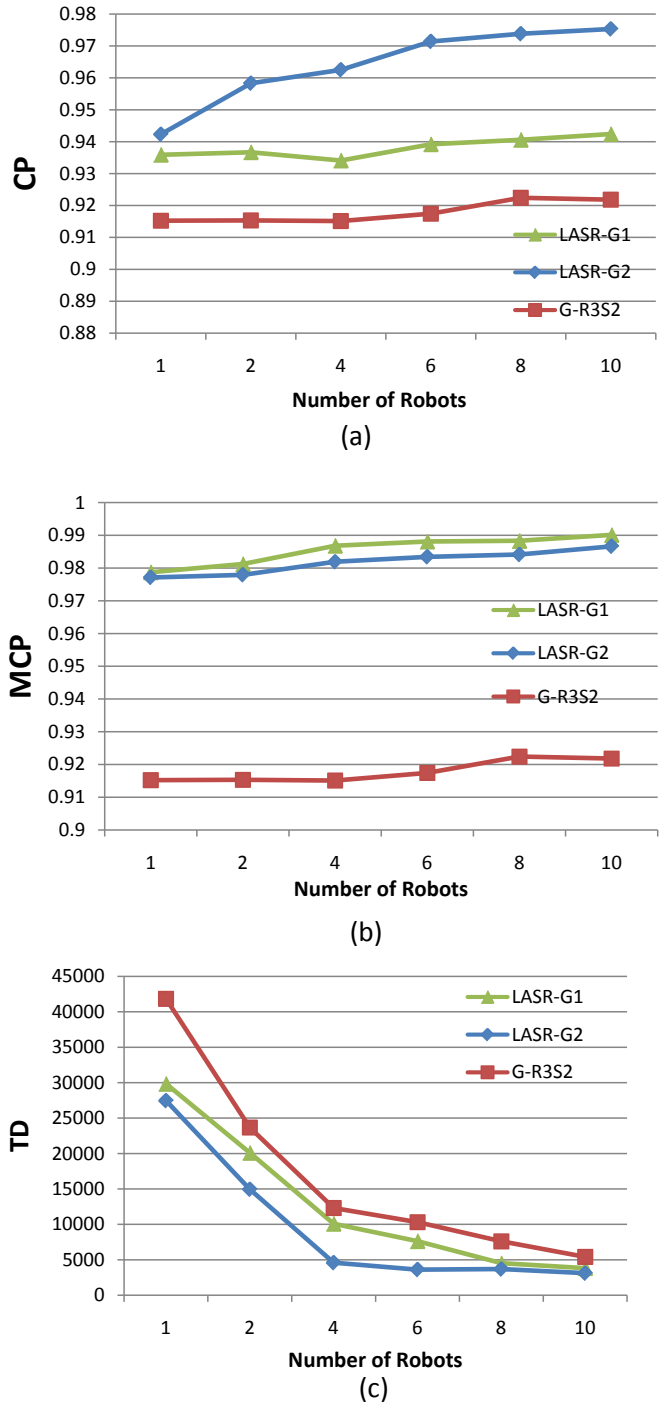


Figure 4.7: Impact of robot number over CP, MCP and TD. (a) measures CP; (b) is MCP indicating maximal coverage algorithms can get; (c) shows robot TD for three algorithms.

result indicates that when taking same TD, LASR-G2 repairs WSN and improves CP faster than other two algorithms.

Figure 4.7(b) shows MCP values three algorithms can meet. Stopping criteria for LASR-G1 and LASR-G2 are set when coverage improvement is less than 0.005 within 500 steps. It shows LASR-G1 can reach highest MCP since it is designed to always pick sensors with most overlap in local area, regardless the distance that would taken for robot to reach the targeted sensor. LASR-G2 has lower value since it needs to balance TD and CP all the time when choosing targeted sensors. For competitor algorithm G-R3S2, since it only consider relocating spare sensors, the performance performance on CP and MCP is limited because of the finite number of spare sensors in ROI, especially for the situations fewer sensors deployed randomly at initialization state.

Figure 4.7(c) compares TD among three algorithms. For fairness, the stopping criterion for LASR-G1 and LASR-G2 are set when corresponding CP values reach stopping value for G-R3S2. Average TD value for LASR-G2 is always shorter than other two at all time for tested scenarios. With only spare sensors defined as targeted sensors, robots take longer TD to explore and pick up spare sensors in G-R3S2. It confirms that by reducing overlapping among sensing range, it can reach same coverage improvement with less TD for robot detecting and moving spare sensors.

4.3.4 Impact of Number of Sensors

We evaluate the impact of number of sensors n on relocation performance for LASR-G1, LASR-G2 and G-R3S2. The simulation analysis is shown in Figure 4.8. We set the number of robots m to 4. The stopping criterion for G-R3S2 is either there is no spare sensor detected or RCP value reaches 0.99.

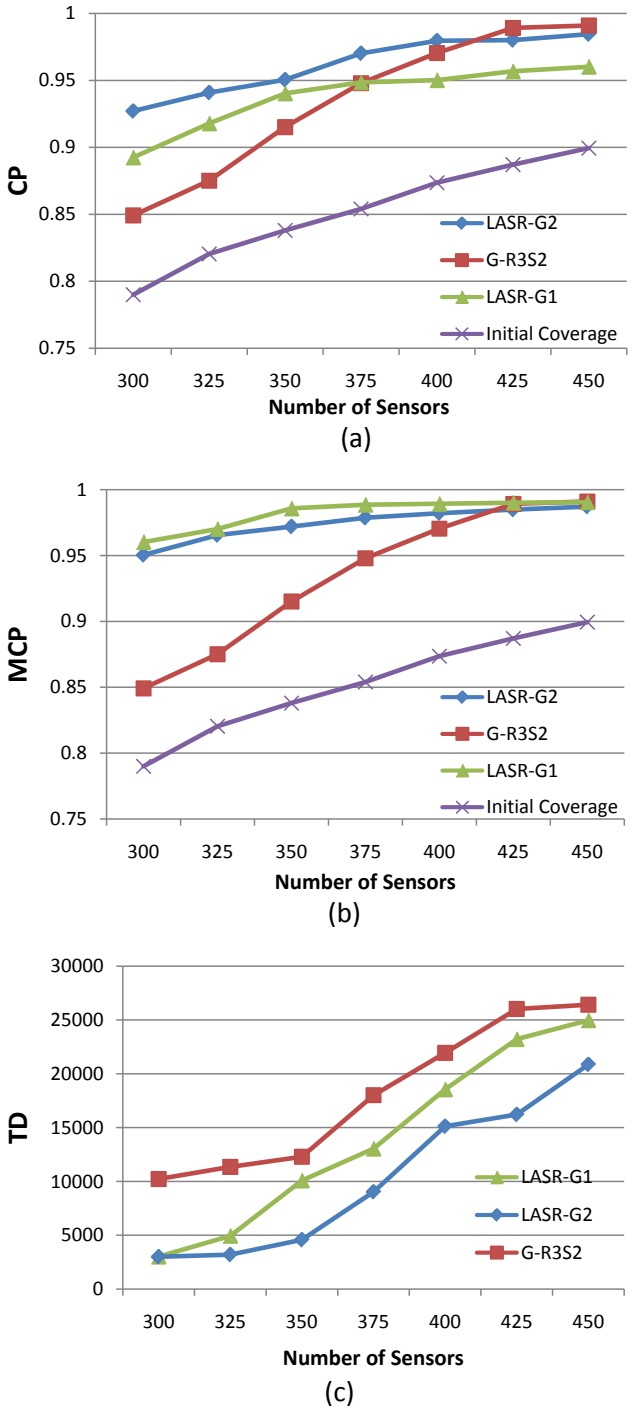


Figure 4.8: Impact of sensor number over CP, MCP and TD. (a) measures CP; (b) is MCP indicating maximal coverage algorithms can get; (c) shows robot TD for three algorithms.

Figure 4.8(a) shows the relation between CP and number of sensors n. We compare CP value when three algorithms reach same TD value for each robot when meeting stopping criterion for G-R3S2. Because of overlapping reduction and limited spare sensors in ROI, LASR-G2 outperforms the other two when n is restricted. However, with the increment of n (e.g., $n > 400$), number of spare sensors in tested region increases, leading to a better performance of CP value for G-R3S2.

Figure 4.8(b) illustrates the relation between MCP value and number of sensors n deployed in tested region. Both LASR-G1 and LASR-G2 stop when coverage improvement is less than 0.005 in 500 steps. With the increase of n, both the random initial coverage and MCP would rise. A stable trends in the curve for LASR-G1 and LASR-G2 indicate the performance is not highly related to n, comparing with G-R3S2 which presents an obvious increase of curve with more sensors involved in. When n is limited (e.g., n=300), the growth rate of either LASR-G1 or LASR-G2 on MCP is larger than of G-R3S2. Because of restricted number of spare sensors generated, performance of G-R3S2 is constrained. When n reaches 400, MCP values for three algorithms are mostly same, since it would be easier for robots to detect spare sensors in G-R3S2 than the situations where limited sensors are deployed. Besides, LASR-G1 has higher MCP than LASR-G2, since the former one consider only coverage improvement while latter one chooses targeted sensors based on both CP and TD .

Figure 4.8(c) explains the relation between average TD for each robot and number of sensors n when reaching same CP as G-R3S2 stops. We can see robots take less steps to reach same coverage in LASR-G2 than the other two, because it balances both CP and TD value on traveling within ROI. For scenarios with $n \geq 400$ in tested region, LASR-G1 and LASR-G2 take larger TD, since most part is already well deployed after random node dropping and previous robot-assisted relocation,

causing robot surrounding area in vicinity similar when applying greedy walk. In that case, it takes longer moving distance for robot to find a "comfortable" place for loaded sensor.

4.3.5 Stability Analysis

Stability Analysis is involved for testing proposed LASR-G1, LASR-G2 and competing G-R3S2 on performance over coverage improvement and robot traveling distance. Although greedy walk and *Least Recently Visited* (LRV) policy are applied in proposed and competing algorithms respectively to reduce randomness when walking within tested ROI, performance and stability is still influenced by other aspects, such as topology of initial sensor deployment. Here we test the algorithms and calculate the standard variance of tested merits to analyze stability.

Stability analysis is based on several groups of scenarios, with number of robots m set to 4 and number of sensors n to 300, 350 and 400. *std* is short for standard deviation, while *ave* stands for the average value of tested variable. Standard deviation is calculated by 20 groups of data for each tested scenario. Each time we collecting related data, initial sensor deployment is randomly generated by our simulator without any control. The result of stability of algorithms is listed below as Table 4.1, 4.2 and 4.3.

Sensor n	300		350		400	
Type	std	ave	std	ave	std	ave
LASR-G1	0.0094	0.8925	0.0095	0.9403	0.0108	0.9502
LASR-G2	0.0102	0.9271	0.0092	0.9505	0.0098	0.9796
G-R3S2	0.0104	0.8492	0.0110	0.9151	0.0094	0.9826

Table 4.1: Standard deviation CP for LASR-G1, LASR-G2 and G-R3S2.

From Table 4.1 and 4.2, we can see that the standard deviation of CP and MCP

Sensor n	300		350		400	
Type	std	ave	std	ave	std	ave
LASR-G1	0.0110	0.9602	0.0115	0.9858	0.0102	0.9893
LASR-G2	0.0085	0.9504	0.0098	0.9721	0.0092	0.9821
G-R3S2	0.0104	0.8492	0.0110	0.9151	0.0094	0.9826

Table 4.2: Standard deviation MCP for LASR-G1, LASR-G2 and G-R3S2.

Sensor n	300		350		400	
Type	std	ave	std	ave	std	ave
LASR-G1	184	3023	1142	10091	2035	18562
LASR-G2	173	3002	351	4586	1452	15125
G-R3S2	2482	10235	3530	12307	7492	21953

Table 4.3: Standard deviation TD for LASR-G1, LASR-G2 and G-R3S2.

are smaller in all scenarios with different number of sensors. Thus we can conclude that stability of coverage improvement is not highly influenced by different initial random deployment. Through repeated relocation process, it is feasible to reach coverage value within a small range when meeting stopping criteria for these three algorithm (stopping criteria for testing CP and MCP is introduced earlier).

Different from Table 4.1 and 4.2, standard deviation value in Table 4.3 shows the stability of G-R3S2 is lower compared to LASR-G1 and LASR-G2. Stopping criterion for G-R3S2 is that no spare sensors exist in ROI or the coverage reaches 0.99. Upon three algorithms reaching same final CP, G-R3S2 takes longer average robot TD with larger fluctuations among each single performance, due to its largest standard deviation of TD value. Unlike LASR-G1 and LASR-G2 replacing sensors to reduce overlapping region, G-R3S2 explores the whole WSN searching for spare sensors. In that case, the location of spare sensors will have an strong impact on robot TD value. If after initial random deployment, spare sensors gather near each other, it is more like for robots to detect all spare sensors with less TD. However if spare

sensors scatter across ROI, especially around boundaries of ROI, it will takes longer TD even if *Least Recently Visited* (LRV) path selection policy is applied. Therefore stability of LASR-G1 and LASR-G2 on TD outperform that of G-R3S2.

Chapter 5

Conclusion and Future Work

We study sensor relocation for coverage maintenance in WSN after initial random deployment. Other than replacing failed sensor node in a well deployed WSN, we focus on relocating and optimizing WSN coverage after random sensor node dropping by reducing overlap among sensing range. Inspired from Ant-Clustering methods [31][6], we proposed localized robot-assisted algorithms, LASR-G1 and LASR-G2, which focus on redeploying both spare sensor and some active sensors whose sensing area is significantly overlapped with neighbours.

The difference between LASR-G1 and LASR-G2 is that LAS-G1 only considers coverage improvement, while LASR-G2 combines the weight of coverage improvement and robot traveling distance. This difference of the two proposed algorithms reflects on how robot chooses targeted picking up sensor at each step. For LASR-G1, robot chooses the one with least *coverage ratio* as long as it is within robot's vision (distance between robot and corresponding sensor is less than r_c). However for LASR-G2, robot selects the one based on the value of predetermined fitness functions.

At each location, robot floods "hello" messages to neighbour sensors and calculates location and distance to each neighbour sensor. By following some functions, robot is

able to measure picking up or dropping probability according to its current load status. If unloaded, calculate picking up probability; if loaded, then dropping probability. Robots make picking up and dropping decisions based on the calculated probabilities, which in turn depend on added or lost coverage area by sensor addition or removal. After decision making, greedy walk is applied to guide robots move to a local best location in its vicinity, also according to current load status. Longer random walk is applied to escape from local minima, which indicates the failure of changing robot load status for a preset robot greedy steps.

Simulation evaluation is applied to compare our proposed algorithms and existing solution G-R3S2. We evaluate parameters σ and δ and choose appropriate values for these two parameters. Then we compare single robot and multi robots scenarios for both proposed algorithms. By comparison and some given figures, we can conclude a more desirable performance for multi robots scenario than single robot. We also discuss the impact of number of robots m and number of sensors n on relocation coverage improvement and robot traveling distance, with the comparison to algorithm G-R3S2. At last we test the stability for our proposed algorithms LASR-G1, LASR-G2 and competing G-R3S2 for coverage improvement and robot traveling distance under 20 times of comparison. Through extensive simulations, we validate our algorithms (LASR-G1 and LASR-G2) and compare them to the only existing localized solution G-R3S2.

In future work, we will consider robot and sensor energy consumption and node failure after certain time period. We will analyse scenarios with robot carrying $N > 1$ sensors. Robot decisions will then also depend on the number of loaded sensors. Further, schedules of modification parameters vp and vd can be updated based on neighborhood and hole status. We assume robots have same physical properties in this paper, however one can expend our work using heterogeneous robots and sensors.

We assumed a simplified model of circular area coverage. Future work could consider probabilistic area coverage, using a model such as in [7]. We also assumed that robots do not collaborate among them directly. If robots could communicate to each other, they may exchange local information and their tasks, using, for example, auction like method [26]. This is an interesting area for future extension of our algorithm.

References

[1] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[2] Maxim A Batalin and Gaurav S Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. *Telecommunication Systems*, 26(2-4):181–196, 2004.

[3] Maxim A Batalin and Gaurav S Sukhatme. The analysis of an efficient algorithm for robot coverage and exploration based on sensor network deployment. In *Robotics and Automation*, pages 3478–3485. IEEE, 2005.

[4] Urszula Boryczka. Ants and multiple knapsack problem. In *Computer Information Systems and Industrial Management Applications*, pages 149–154. IEEE, 2007.

[5] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless networks*, 7(6):609–616, 2001.

[6] Ling Chen, Xiaohua Xu, Yixin Chen, and Ping He. A novel ant clustering algorithm based on cellular automata. In *Intelligent Agent Technology*, pages 148–154. IEEE, 2004.

[7] Pingsheng Chen and Weidong Hu. Sleepcwake up scheduling with probabilistic coverage model in sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 29(1):1–16, 2014.

[8] Marco Dorigo, Gianni Di Caro, and Luca M Gambardella. Ant algorithms for discrete optimization. *Artificial life*, 5(2):137–172, 1999.

[9] Rafael Falcon, Xu Li, Amiya Nayak, and Ivan Stojmenovic. The one-commodity traveling salesman problem with selective pickup and delivery: An ant colony approach. In *Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010.

[10] Rafael Falcon, Xu Li, Amiya Nayak, and Ivan Stojmenovic. A harmony-seeking firefly swarm to the periodic replacement of damaged sensors by a team of mobile robots. In *Communications (ICC)*, pages 4914–4918. IEEE, 2012.

[11] Greg Fletcher, Xu Li, Amiya Nayak, and Ivan Stojmenovic. Back-tracking based sensor deployment by a robot team. In *Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9. IEEE, 2010.

[12] Greg Fletcher, Xu Li, Amiya Nayak, and Ivan Stojmenovic. Randomized robot-assisted relocation of sensors for coverage repair in wireless sensor networks. In *Vehicular Technology Conference Fall*, pages 1–5. IEEE, 2010.

[13] Antoine Gallais, Jean Carle, David Simplot-Ryl, and Ivan Stojmenovic. Localized sensor area coverage with low communication overhead. *Mobile Computing*, 7(5):661–672, 2008.

[14] Hipólito Hernández-Pérez and Juan-José Salazar-González. The one-commodity pickup-and-delivery travelling salesman problem. In *Combinatorial Optimization, Eureka, You Shrink!*, pages 89–104. Springer, 2003.

[15] G Indirani and K Selvakumar. A swarm-based efficient distributed intrusion detection system for mobile ad hoc networks (MANET). *International Journal of Parallel, Emergent and Distributed Systems*, 29(1):90–103, 2014.

[16] Xu Li, Greg Fletcher, Amiya Nayak, and Ivan Stojmenovic. Randomized carrier-based sensor relocation in wireless sensor and robot networks. *Ad Hoc Networks*, 2012.

[17] Xu Li, Greg Fletcher, Amiya Nayak, and Ivan Stojmenovic. Placing sensors for area coverage in a complex environment by a team of robots. In *ACM Transactions on Sensor Networks*. ACM, to appear.

[18] Xu Li, Amiya Nayak, David Simplot-Ryl, and Ivan Stojmenovic. Sensor placement in sensor and actuator networks. *Wireless Sensor and Actuator Networks: Algorithms and Protocols for Scalable Coordination and Data Communication*, pages 263–294, 2010.

[19] Xu Li, Nicola Santoro, and Ivan Stojmenovic. Mesh-based sensor relocation for coverage maintenance in mobile sensor networks. In *Ubiquitous Intelligence and Computing*, pages 696–708. Springer, 2007.

[20] Xu Li, Nicola Santoro, and Ivan Stojmenovic. Localized distance-sensitive service discovery in wireless sensor and actor networks. *Computers*, 58(9):1275–1288, 2009.

[21] Wen-Hwa Liao, Yucheng Kao, and Ru-Ting Wu. Ant colony optimization based sensor deployment protocol for wireless sensor networks. *Expert Systems with Applications*, 38(6):6599–6605, 2011.

[22] Xuxun Liu and Desi He. Ant colony optimization with greedy migration mechanism for node deployment in wireless sensor networks. *Journal of Network and Computer Applications*, to appear.

[23] Erik D Lumer and Baldo Faieta. Diversity and adaptation in populations of clustering ants. In *The third international conference on Simulation of adaptive behavior: from animals to animats 3: from animals to animats 3*, pages 501–508. MIT Press, 1994.

[24] Kalypso Magklara, Dimitrios Zorbas, and Tahiry Razafindralambo. Node discovery and replacement using mobile robot. In *Ad Hoc Networks*, pages 59–71. Springer, 2013.

[25] Yongguo Mei, Changjiu Xian, Saumitra Das, Y Charlie Hu, and Yung-Hsiang Lu. Sensor replacement using mobile robots. *Computer Communications*, 30(13):2615–2626, 2007.

[26] Ivan Mezei, Veljko Malbasa, and Ivan Stojmenovic. Greedy extension of localized auction based protocols for wireless actuator task assignment. *Adhoc & Sensor Wireless Networks*, 17(1-2):73–85, 2013.

[27] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[28] Yipeng Qu and Stavros V Georgakopoulos. A centralized algorithm for prolonging the lifetime of wireless sensor networks using particle swarm optimization. In *Wireless and Microwave Technology Conference (WAMICON)*, pages 1–6. IEEE, 2012.

[29] Yipeng Qu and SV Georgakopoulos. Relocation of wireless sensor network nodes using a genetic algorithm. In *Wireless and Microwave Technology Conference (WAMICON)*, pages 1–5. IEEE, 2011.

[30] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[31] André L Vizine, Leandro Nunes de Castro, Eduardo R Hruschka, and Ricardo R Gudwin. Towards improving clustering ants: An adaptive ant clustering algorithm. *Informatica (Slovenia)*, 29(2):143–154, 2005.

[32] Guiling Wang, Guohong Cao, Tom La Porta, and Wensheng Zhang. Sensor relocation in mobile sensor networks. In *INFOCOM Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2302–2312. IEEE, 2005.

[33] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12):2292–2330, 2008.

[34] Mohamed F Younis, Sookyoung Lee, and Ameer Ahmed Abbasi. A localized algorithm for restoring internode connectivity in networks of moveable sensors. *Computers*, 59(12):1669–1682, 2010.