# Market-Based Sensor Relocation by a Team of Robots in Wireless Sensor Networks

by

Haotian Li

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 supervisors, Prof. Amiya Nayak and Prof. Ivan Stojmenovic for his exceptional guidance and support not only in this thesis but also in my study and research. Without their help, I cannot finish this thesis. Special thanks to Dr. Cheng Wang who gave me precious advices and suggestions.

I wish to express my cheers to my friend Yuan Wang who always helps and encourages me.

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

**Abstract**

Randomly scattered sensors may cause sensing holes and redundant sensors. In carrier-based sensor relocation, mobile robots (with limited capacity to carry sensors) pick up additional or redundant sensors and relocate them at sensing holes. In the only known localized algorithm, robots randomly traverse field and act based on identified pair of spare sensor and coverage hole. We propose a *Market-based Sensor Relocation* (MSR) algorithm, which optimizes sensor deployment location, and introduces bidding and coordinating among neighboring robots. Sensors along the boundary of each hole elect one of them as the representative, which bids to neighboring robots for hole filling service. Robot randomly explores by applying Least Recently Visited policy. It chooses the best bid according to *Cost over Progress ratio* and fetches a spare sensor nearby to cover the corresponding sensing hole. Robots within communication range share their tasks to search for better possible solutions. Simulation shows that MSR outperforms the existing competing algorithm G-R3S2 significantly on total robot traversed path and energy, and time to cover holes, slightly on number of sensors needed to cover the hole, and the cost of additional messages for bidding and deployment location sharing.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background Information

### 1.1.1 What is WSN?

Wireless sensor network (WSN) is a typical technology that can be widely used in national defense, national security, environmental science, traffic management, disaster prediction, health care, manufacturing and some other areas. Wireless sensor network is composed of a lot of uniform or different nodes. Each node consists of a sensor data acquisition module collecting information such as temperature, humidity or some other types of data. Recent development in Micro-Electro-Mechanical Systems (MEMS) technology provides possibility for miniaturization of sensors. This further promotes the intelligence of sensors. The integration of MEMS technology and radio frequency (RF) communication technology brings development of wireless sensor network.

In sensor network applications, the sensor nodes are usually placed at locations without any infrastructure. The position of sensor nodes can not be precisely set in advance. The mutual position of neighbor nodes is not known either. For example, a

large number of sensor nodes could be scattered by plane to the vast area of primeval forest, or randomly placed into dangerous area that human beings can not reach. This requires sensor nodes have the ability to self-organize, automatically configure and manage themselves. These sensor nodes form a multi-hop wireless network system through topology control mechanisms and network protocols to collect data or monitor in targeted area.

There are also some applications for our everyday life. For instance, Mitsubishi Electric has developed a small low-power wireless sensor network module, which can be used to build a specific low-power wireless ad hoc network. This is designed to replace the current dedicated home security network. Specifically, some applications use infrared sensors to detect whether there is anyone nearby. Besides, some use acceleration sensor to detect vibration of window and furniture.

### 1.1.2 Components in WSN

The targeted area is defined as *region of interest* (ROI). As sensors in ROI usually provide monitoring service, each of them provides monitored area. Usually it is a circle area and the circle area is called *sensing circle*. The radius of the circle is defined as *sensing range* $r_s$ [15, 17, 28]. Fig. 1.1 shows an example of sensor's sensing circle and sensing range. The sensor $S$ can collect specific types of information within the sensing circle.

The maximum distance for two sensors to communicate with each other is defined as *communication range* $r_c$. In other words, sensor can talk to another sensor if the distance between these two is shorter than $r_c$. For each sensor in ROI, other sensors within communication range are its *one-hop neighbors*.

The continuous area not covered by any sensors is called *sensing hole* or *coverage hole*. Sensors on the boundary of a sensing hole is called *boundary sensor* or *boundary*

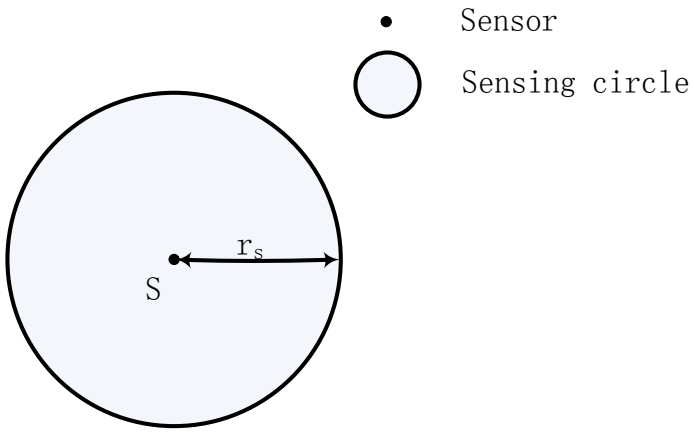



Figure 1.1: An example of sensor's sensing circle

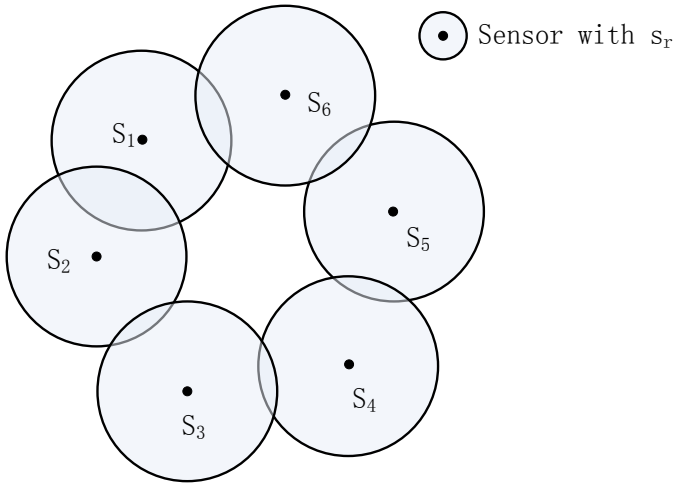



Figure 1.2: An example of sensing hole

*node*. For example, Fig. 1.2 shows the area in the enclosed by $S_1 \sim S_6$ is a sensing hole. $S_1 \sim S_6$ are boundary nodes.

Sensors redundant for providing coverage is called *spare sensor* or *redundant sensor*. The sensing area of spare sensor is fully covered by its one-hop neighbors. Spare sensors sleep to save energy until it is relocated to provide coverage. Non-redundant sensors must remain *active* for coverage, so that they are called *active sensors*. In Fig. 1.3, $S_6$'s sensing area is covered by $S_1 \sim S_5$, so $S_6$ is a spare sensor. The removal of $S_6$ does not affect the coverage ratio in ROI. $S_1 \sim S_5$ are active sensors. If any one

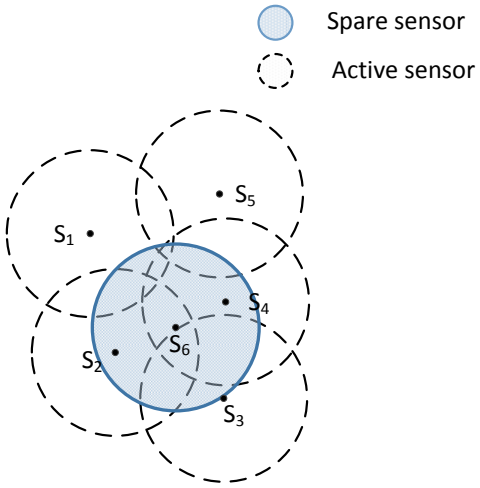



Figure 1.3: An example of spare sensor and active sensor

of them is removed, a sensing hole will be generated.

### 1.1.3 Solutions to Achieve High Coverage

Deploying sensors at desired location manually is huge work for human beings. Some algorithms use robot with mobility to deploy sensors in region of interest[10, 16, 3, 4]. Thus, wireless sensor and robot network (WSRN) is introduced. However, due to the large scale of sensor network, deploying sensors with machine is not efficient.

In most cases, sensors are randomly scattered in area of interest. As randomly deployed sensor in the network cannot provide good coverage in some cases, some sensors in the environment need to be relocated to provide better coverage. Some research works introduce mobile sensors in the network. They can adjust their locations to fill reported sensing holes to improve coverage[18, 6, 29, 19, 17]. In self-deployment approaches, when there is a sensing hole generated by node failure or uneven pre-deployment, mobile sensors move to uncovered area to fill coverage hole. Mixed sensor network is composed of mobile sensors and static sensors. Static sensors are randomly deployed to provide coverage and mobile sensors move to detected

sensing holes to improve coverage. When static sensors detect a sensing hole nearby, they report to a mobile sensor. This mobile sensor then move to indicated place to fill the sensing hole.

Equipping locomotion on all sensors or a set of sensors makes high hardware cost. In order to reduce the cost of coverage repair, some solutions use robot with certain load capacity to adjust the locations of sensors in ROI to improve coverage. As sensors are randomly scattered, the density is uneven for each part of region. In some area, sensor density is so high that some sensors are redundant for providing coverage. This kind of sensors can be collected by robot to fill coverage holes. Several centralized approaches [9, 24, 25] introduce a base station that keeps in contact with all sensors and robots in the environment all the time. With knowledge of the whole topology, base station assigns proper robot to fill the reported sensing holes efficiently. There are also several localized solutions [11, 15, 27] presented for carrier-based sensor relocation. Sensors periodically broadcast "Hello" message to inform their one-hop neighbors of their existence. When sensors are aware of node failure or sensing holes, they report to robot so that robot could move a redundant sensor to an uncovered place to fill the sensing hole.

In WSRN, robots can take sensors as payload and adjust sensors' location to get better coverage. Sometimes, robot's cooperation improves the performance of algorithms (e.g., save time or robot's energy). Several robot task allocation approaches are proposed [13, 21, 26].

## 1.2 Problem Statement

We assume sensors are randomly deployed in region of interest (ROI), without guarantee of satisfactory coverage. Due to stochastic node dropping, coverage holes and redundant sensors (whose sensing range is fully overlapped with neighbouring sensors)

are generated, with redundant sensors used to fill coverage holes. In some existing algorithms, locomotion is involved in sensor relocation, such as mobile sensor self-relocation, and carrier-based sensor relocation, to improve coverage. In mobile sensor scenarios, after randomly scattered in ROI, mobile sensors change their positions to improve coverage [16]. However, considering the large scale of network, the cost is very high to equip locomotion for every sensor. In that case, robot is introduced to migrate in ROI and relocate sensors[15]. To reduce the budget on hardware, we use a small team of robots for coverage enhancement.

## 1.3 Existing Solutions

There are several solutions for carrier-based coverage repair or sensor deployment [3, 16, 22] in the literature. Robots explore ROI and deploy sensor if needed. In [3] and [16], robot makes serpentine movement and deploys sensors in ROI. Robot moves for a certain distance which is related to sensor's sensing range, and detects whether there is a sensing hole. When robot encounters coverage hole, it deploys a sensor and then continues moving according to the moving strategy. Several moving strategy is presented in [3] to recover from the obstacle. However, robot may come to a dead end that all the pre-predefined directions are blocked either by obstacles or deployed sensors. [16] is an improved algorithm based on [3]. When robot makes serpentine movement and deploys sensors to fill coverage holes, it records the last sensor that near a sensing hole. Robot recovers from a dead end by marching to the last sensor and continue deploy sensors their.

Mei et al. present three protocols for coverage repair problem [22]. The first one is centralized. Robot is acting as both manager and carrier. Sensor failure is detected by neighbors of the failed node. Sensors report detected node failure to robot. Robot will come to replace the failed sensor with a new one. The remained two are fixed

distributed algorithm and dynamic distributed algorithm. In this two algorithms, the whole ROI is divided into several subareas according to Voronoi Diagram and there is one robot in each subarea acting as manager and carrier. The border of different subarea is fixed in fixed distributed algorithm, while it is changeable in dynamic distributed algorithm according to robots' locations.

[16] and [22] both assume infinite capacity of robot, which is not feasible for robot to carry set of sensors for entire network. In some cases, some already deployed sensors are redundant for coverage and can be reused for robot to fill coverage holes for an economic point of view, which can be defined as sensor relocation problem.

One centralized solution for sensor relocation coverage repair problem is brought in [9]. The coverage repair problem is treated as NP-complete Travelling Salesman Problem (TSP) combined with Ant Colony Optimization (ACO). It assumes a lot of pick up customers (i.e., redundant sensor) are available and robot can pick up and deliver to desired places (i.e., sensing hole). [22, 9] only consider sensing hole generated by node failure, so that it's treated as point rather than area. Robot only needs to place a sensor at location of the failure sensor. However, in some cases, coverage hole may also be caused by initial topology, which is not considered in these algorithms.

There are several solutions for carrier-based coverage repair [16, 22] in the literature. Robots explore ROI and deploy sensor. [16] and [22] both assume infinite capacity of robot, which is not feasible for robot to carry set of sensors for entire network. In some cases, already deployed sensors can be reused for robot to fill coverage holes. One centralized solution for sensor relocation coverage repair problem is brought in [9]. The coverage repair problem is treated as NP-complete Travelling Salesman Problem (TSP) combined with Ant Colony Optimization (ACO). [22][9] only consider sensing hole generated by node failure, in which sensor replacement

happens at exact same location of failed nodes. However, in some cases, polygon-shaped coverage hole may also be caused by initial topology.

To the best of our knowledge, R3S2 [15] is the only localized solution for carrier-based sensor relocation applicable for team robots. Sensors locally identify nearby sensing holes. Robots move randomly in ROI and detect sensing holes and redundant sensors. If a robot is currently carrying sensors, it deploys loaded sensor at nearest sensing hole rather than considering local maximal possible coverage improvement from robot's vision. Since in most cases,robot can only obtain information of part of coverage hole in R3S2, selected deployment location has limited performance to improve coverage ratio.

## 1.4 Motivations and Objectives

As an alternative to using sensor deployment algorithm, we expect to relocate the location of sensor after the sensors are randomly deployed. Some previous solutions equip mobility to all sensors in WSN, but the hardware cost is pretty high. One alternative is to use some carriers to help the sensor relocation. A solution for carrier-based sensor relocation problem should be designed localized to improve the scalability and robustness of the algorithm. In addition, sensors should locally identify shape of sensing holes nearby. To be optimized, we consider number of messages transmitted and distance robot travel as measurement to be minimal.

Previous algorithms mostly consider sensing hole as one point which can be covered by one sensor which assumes sensors are deployed to fully cover certain area and some sensors fails which is needed to be fixed. This is different from the problem we focus on.

Robots in R3S2 move randomly in the environment fetch redundant sensors and deploy them at sensing holes. Robots always fill the nearest sensing hole without

considering the progress made. This deploy strategy is not efficient, which causes high cost in the whole process.

We describe a new localized algorithm which solve carrier-based sensor relocation problem. This algorithm is expected to outperform the existing sensing relocation algorithm on repair latency message cost and robots short travel distance. In addition, our algorithm is expected to achieve a full coverage.

Inspired by incompleteness and limitation of previous works, we address a practical and efficient algorithm involving robot collecting redundant sensors to fill coverage holes.

## 1.5 Assumptions

We consider identical sensors randomly scattered in ROI. Robots are used to fetch spare sensors in ROI and relocate them at sensing holes. They have unique ID and same communication range $R_c$. Robot can carry only one sensor at one time. Sensors and robots are aware of their locations with help of GPS. Robot moves in the same speed and has unlimited energy. Message delay from one sensor to another sensor is set to the same since sensors communication range is not long enough to be comparable to transmission speed. We consider ideal MAC and physical layer. There is no collision or failure in packet transmission. In other words, all messages can be delivered to the destination finally.

## 1.6 Our Contribution

Inspired by incompleteness and limitations of previous work, we propose a practical and efficient localized *Market-based Sensor Relocation* (MSR) algorithm where robot collects redundant sensors to fill coverage holes. Robot adopts Least Recently Vis-

ited (LRV) policy[2] to explore in ROI, which is the same as applied in [15]. Each robot applies LRV independently on other robots. All sensors are assumed to be identical, and locally detect nearby sensing holes by checking uncovered arcs on their sensing perimeters. Spare sensors are identified by using the algorithm in [12] and are reported to robot within communication range. Sensors on the boundary of a hole cooperate to identify the shape of the sensing hole and choose a representative sensor to collect coverage hole information. Representative tests many possible locations inside coverage hole, to reduce the number of arcs surrounding coverage hole, and to minimize coverage overlap with originally deployed sensors. The representative then relays selected best location to boundary nodes along the hole. Boundary nodes bid to robot within their communication range with possible deploying location and newly covered area. Receiving more than one bids, the robot considers the *Cost over Progress*(CoP) ratio and chooses the best bid and corresponding redundant sensor and deployment location. Cost is the distance robot travels for fetching a redundant sensor and marching to fill the sensing hole. The progress is the newly covered area. By considering CoP of many possible deployment locations, robot tends to choose the location with high coverage progress and low travel distance. Communication among robots enlarges robots' vision so that they have more choices of spare sensor and sensing holes. Through extensive simulation, we compare the performance of MSR with the only existing solution R3S2[15]. Coverage ratio of MSR is higher than G-R3S2 by approximately 5% with the same number of sensors. In high coverage ratios, MSR saves around 70% of robot energy compared to G-R3S2, and time consumed for relocation process in MSR is over 60% less than that in G-R3S2. Sensors' message cost in MSR is lower than R3S2 in most cases.

## 1.7 Organization

The paper is organized as follows. Chapter 2 reviews existing solutions for works related to sensor relocation problem in WSN. After that, we introduce MSR in Chapter 3. Our extensive simulation is shown in Chapter 4. We draw the conclusions in Chapter 5.

# Chapter 2

# Literature Review

## 2.1 Carrier-based Sensor Deployment

### 2.1.1 Least Recently Visited Approach

Batalin et al. presented a carrier-based algorithm called Least Recently Visited (LRV) [2]. In this algorithm, robot doesn't need to have the information of the whole environment nor the ability of localize itself. Robot can detect boundaries and obstacles, and sensors have the ability of calculation and counting. When the robot meets a sensor, sensor gives suggestion to the robot which direction to go. The suggestion depends on which directions the robot followed recently. Each sensor has a piece of memory keeping the times of every direction to which the robot forward recently. Each time the robot arrives to a sensor or take a sensor's suggestion and head in a certain direction, the counter of this direction would increase by one. When the robot cannot detect messages sent by the former sensor, it deploys a sensor there. In this way, all the sensors keep the record of the number of direction which is followed by the robot. Although this algorithm could achieve a full coverage, the author fails to present a termination rule.

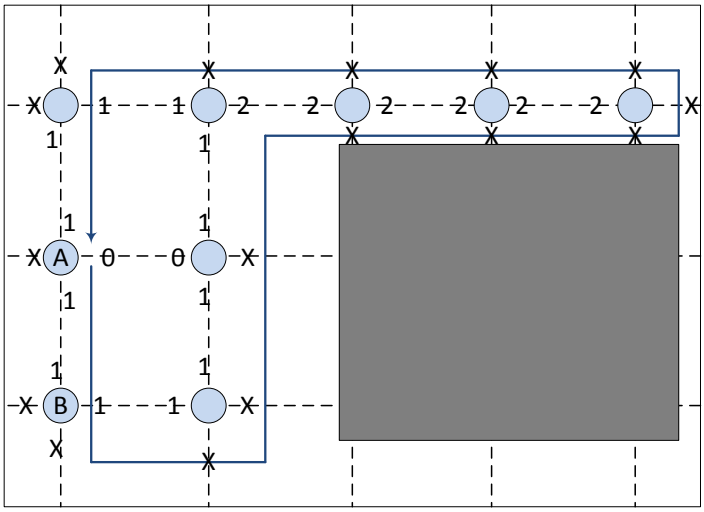



Figure 2.1: Least Recently Visited (LRV) Algorithm

Fig. 2.1 shows us how LRV works. The robot starts at A and begins to deploy sensors according to the suggestions of the sensors and returns to A. The numbers indicate how many times robot visits a grid from each direction. As no terminating rule is presented, this robot will move in this circumstance forever.

### 2.1.2 Back Tracking Approach

Li et al. proposed Back Tracking Deployment (BTD) [16] algorithm which improves the performance of LRV in some aspects. In this algorithm, the robot makes serpentine movement and deploys sensors all the way. Robot is assumed to have the ability to localize itself and detect boundaries. At the same time, the sensors need a small piece of memory to keep track of their back tracking pointers. The back tracking pointers point to the last sensors around which there are empty grid points. This enables BTD to recover from a dead end resulted from obstacles. As robot knows the sensor near sensing hole, it marches directly to the sensor and deploy sensors to cover sensing hole.

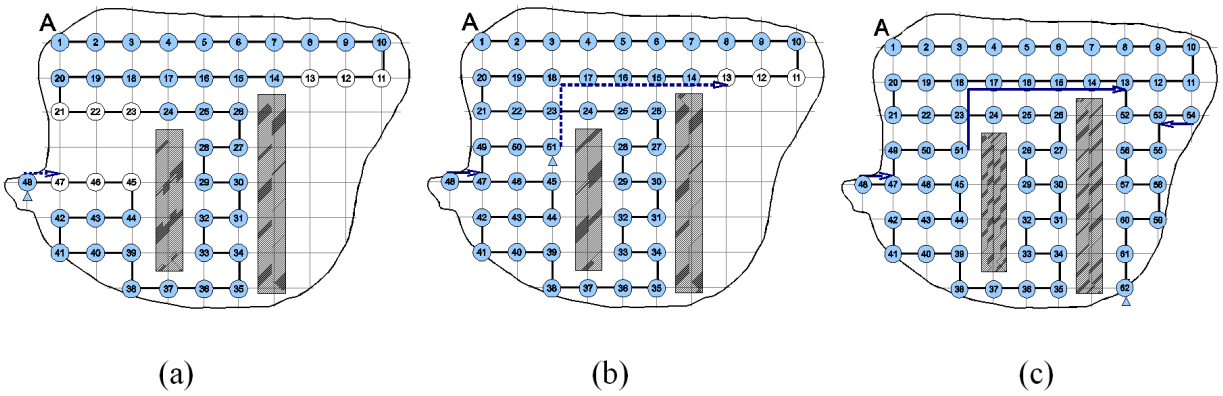



Figure 2.2: An example for BTD (a) Dead End, (b) Shortcut and (c) Full Robot Trajectory

As we can see from the Fig. 2.2, the robot begins to deploy sensors from the northwest corner of the area of interest. As the deployment continues, the ID of the sensors increases. Every sensor back tracks to the last white sensor which has a smaller identification. For example, $S_1 \sim S_{11}$ don't have back tracking sensors and $S_{12}$ and $S_{13}$ back track to $S_{11}$ and $S_{12}$ separately. When the robot deploys $S_{48}$, it reaches a dead end. However, with the help of its back tracking pointer which points to $S_{47}$, it recovers from the dead end and achieve a full coverage. Moreover, the author also presents a solution with team robots and also the maintenance of the network. However, BTD deals with sensor deployment rather than sensor relocation problem where redundant in ROI can also be used to achieve extra coverage.

## 2.2 Self Relocation

### 2.2.1 Cluster Approach

Kehdr et al. present a redundant sensor discovery algorithm and a relocation algorithm [14]. Redundant sensors are not useful for coverage providing, so they can be

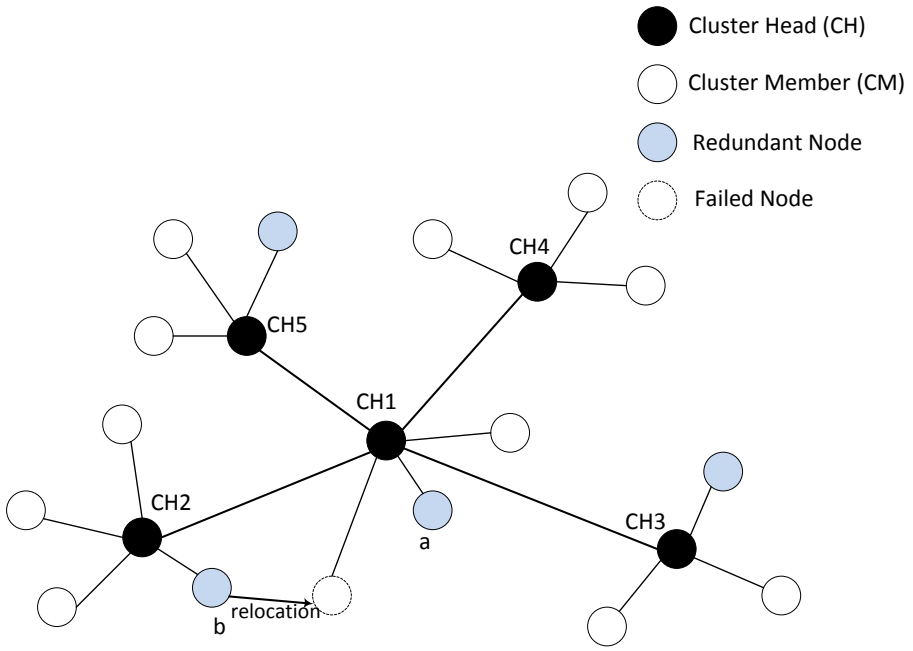



Figure 2.3: Cluster-Based Network

used to help to fill sensing holes. All of the sensors are mobile ones and they can move to a sensing hole by their own if necessary. Every sensor nodes have same characteristics, and communication range is twice larger than sensing range. The authors assume that every sensor knows its own location and all nodes are organized as clusters which are managed by cluster heads. In this algorithm, the sensing area of sensors is divided into equal six sectors. If all sectors are covered by neighbor nodes, the sensor is said to be redundant. For each sector, if at least one sensor locates in a sector, the sector is considered to be covered. Otherwise, sector would check whether all its boundary point is covered by another sensor. If it still cannot be covered, the sector would be divided into smaller part and the small parts continue check whether can be covered.

The sensor relocation algorithm is based on clusters. Each cluster has a head called cluster head (CH), and CH can communicate with its neighbouring CHs. Information

about failure node and spare sensor can be reported to the CH in each cluster. Then the CH would find a spare sensor in the cluster it manages which is nearest to the failure node. At the same time, the CH gets information of spare sensor belonging to CH neighbours' clusters. If there is a spare sensor which is nearer to the failure node, this sensor is assigned to take care of the sensing hole.

Fig. 2.3 illustrates a cluster based network from which we can see CH1 detects there is a sensing hole and $a$ is a spare sensor in its cluster. Then CH1 receives message from its CH neighbour CH2 that $b$ is also a spare sensor. After calculation, CH1 finds $b$ is nearer to the failed node, so $b$ is going to move to replace failed node.

### 2.2.2 Bidding and Proxy Approach

Wang et al. [28] present a price-based sensor self-relocation solution that mobile sensors relocates to sensing holes to increase coverage. All static sensors have same sensing range and can locate themselves. Besides, mobile sensors can plan their path from current location to a desired destination. Static sensors can detect sensing holes and mobile sensors can act as hole healing server. The first issue is how to detect sensing hole and how to measure the size of the hole. The authors use Voronoi Diagram to detect sensing hole. Sensors only take their one hop neighbor into consideration and divide the whole area into several subareas and each sensor takes charge of on area. In each subarea, the sensor decides whether its covering circle has covered the whole subarea. If it is not, there is sensing holes. Specifically, if the distance from sensor to the farthest Voronoi vertex is larger than sensing range, there would be a sensing hole.

Every mobile sensor has a base price which is the uncovered area caused by absence of the sensor. Static sensors are bidders of the coverage hole healing services. Their bids are the estimated sizes of the hole they detect. Static sensors bid mobile sensors

that have a base price lower than their bid. Mobile sensors choose the highest bid and move to the target locations provided by static sensors. They always choose farthest Voronoi vertex as target because this enables the newly covered area to be maximum. After the service advertisement process, each static sensor has a list of mobile sensors and has to choose which sensors to bid. The authors present two criteria: distance-based and price-based. In distance-based approach, sensor chooses the nearest mobile sensor to bid, while in price-based approach, sensor chooses the cheapest price to bid.

Then if the subarea in a sensors Voronoi area needs to be repaired, the sensor bid to mobile sensors to come to repair the hole. The sensor estimates the area could be covered by the new sensor and the cost is the distance mobile sensor travels. Mobile sensors choose one best bid which has a largest ratio of area to distance to take and move to repair the specific hole.

Besides, authors also propose a proxy-based bidding protocol. This protocol improves in terms of energy efficiency and load balance. To avoid significantly increasing communication overhead, authors use proxy sensor which is nearest to virtual location of mobile sensors. The proxy sensor advertises service and process bidding messages. When mobile sensor receives first bidding message, it replies a message to the bidder, then the bidder act as this mobile sensors proxy. The proxy waits for other bidding messages for the mobile sensor. After several rounds, the proxy sends the final best bid back to the mobile sensor, and the mobile sensor comes to fix certain hole.

### 2.2.3 Quorum Approach

Wang et al. propose a Grid-Quorum solution [29] to find redundant mobile sensors with transmitting small number of messages. Also, the algorithm applies cascaded movement to relocate the redundant sensors at expected location. The whole area is divided into several parts and in each part there is a head sensor collecting information

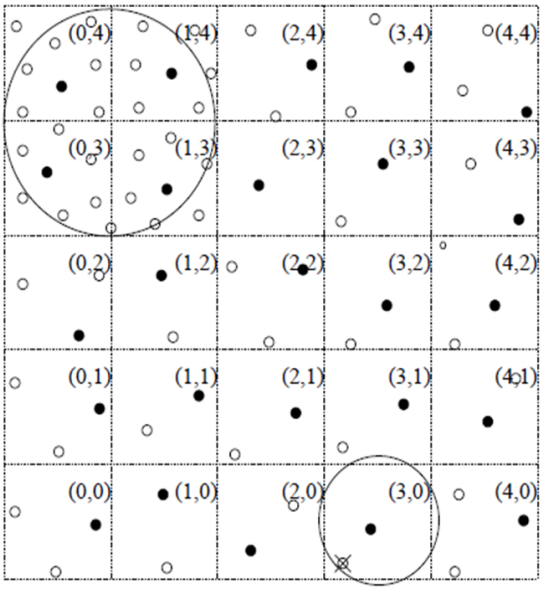



Figure 2.4: The system model

in this area. The grid head also monitors its group sensors and initiates a relocation process if new events or sensor failure happens. In the Grid-Quorum solution, the grid heads belong to the grids in a row are organized in a quorum called supplied quorum and grid heads belong to grids in a column are organized in demand quorum. Head of grid that containing redundant sensors broadcasts in supply quorum. If a grid head detects there is a sensing hole within its gird, it searches along the demand quorum. As a supply quorum and a demand quorum must generate an intersection, the grid can always find a grid that contains a redundant sensor.

Fig. 2.4 shows an example of the model. Grids (0, 4), (1,4), (0, 3), (1, 3) have redundant sensors and there is a sensing hole in Grid (3, 0). Grid (3, 0) searches in demand quorum and Grid (1, 0) can reply information of redundant sensor. In this way, number of messages is cut by half comparing to broadcasting. The authors propose a cascaded movement solution to deal with the moving issues. The idea is find some intermediate sensors and use them for relocation in order to reduce the

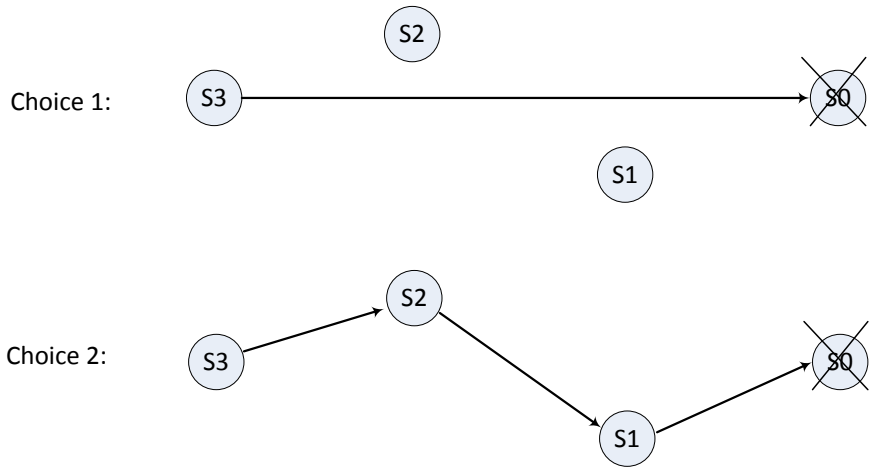



Figure 2.5: Cascaded movement

delay. Fig. 2.5 gives us an example for cascaded movement. $S_3$ is a redundant sensor and $S_0$ is a virtual target node. $S_3$ moves to $S_2$'s location, $S_2$ moves to $S_1$s location, and $S_1$ moves to $S_0$. A distributed protocol is proposed to minimize the difference between total energy consumption and minimum remaining power.

### 2.2.4 Mesh-based Approach

Li et al. proposes a mesh-based sensor relocation protocol (MSRP) [18] for network maintaining purpose in a mobile sensor network. At first, the interested area is fully covered. Sensors may fail at any time for some reasons, so redundant sensors in the network would be used to fix these sensing holes. Some active nodes are selected as proxy by redundant nodes. When a node failure occurs, this nodes neighbor would find a redundant sensor nearby through the mesh-based network.

Fig. 2.6 gives a general idea how MSRP works. In this figure, redundant sensor is represented by colorful small dot, while bid colorful dots are proxies. Proxies broadcast redundant information in four directions: east, west, south and north through colorful links in the figure. After node $a$ fails, its neighbor $s$, $e$, $n$ and $w$ work in

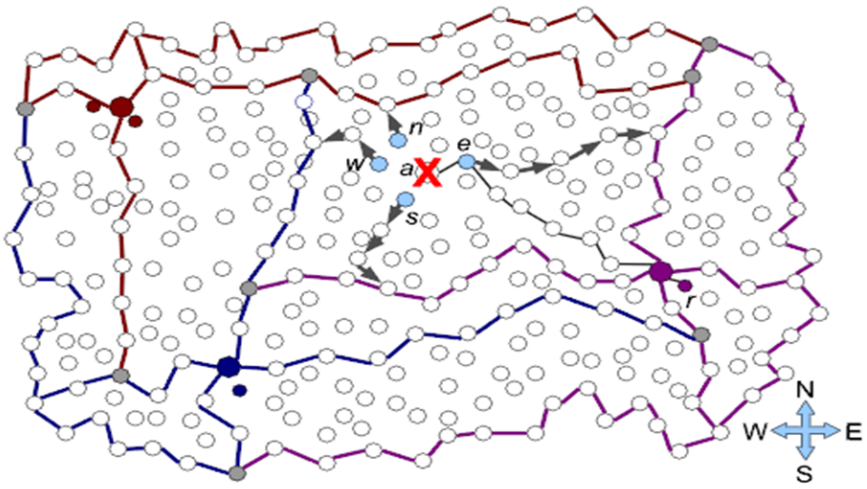



Figure 2.6: A general view about how MSRP works

collaboration to find a redundant sensor to replace $a$. As nodes in the colorful links have knowledge of at least one redundant sensor, when searching message arrives to colorful links, nodes on colorful links can reply with information about an available redundant node. In this way, redundant sensors can be found to fill the sensing hole caused by node failure.

Self-relocation approach includes locomotion for either part of sensors or all sensors leading to high hardware cost for relocation.

## 2.3 Carrier-based Coverage Repair

### 2.3.1 Voronoi Diagram Approach

Mei et al. present three protocols for coverage hole repair problem [22]. In centralized algorithm, a robot functions as a central manager which does not move. All robots and sensors update their information to the manager. If node failure happens, its neighbor report to the manager and then, manager assign a robot to repair it. The second one is fixed distributed manager algorithm. In this algorithm, each robot is

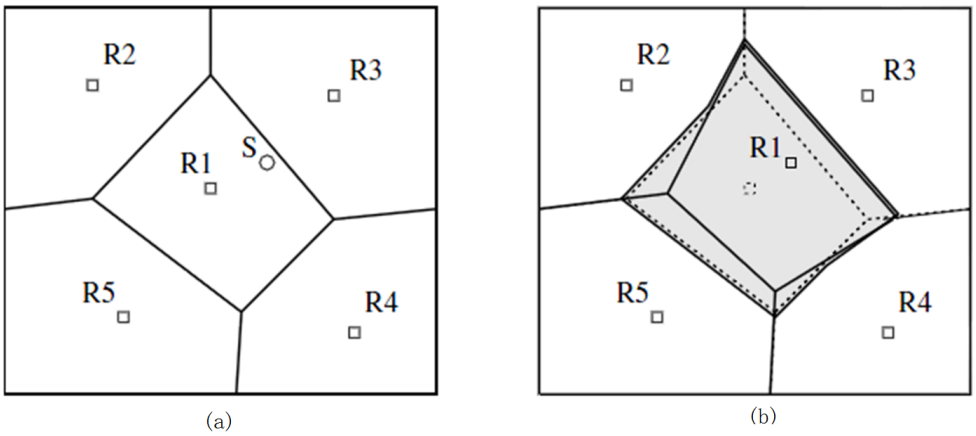



Figure 2.7: An example for Voronoi Diagram

assigned an equal size area. Robots in each area behave as manager and carrier at the same time. Robot and sensors in each subarea runs the same algorithm as centralized algorithm. The last one is dynamic distributed manager algorithm. There is no fixed boundary between two robots. Instead, boundary is constructed dynamically as Voronoi diagrams. As we can see from Fig. 2.7, the whole environment is divided into five parts with one robot in each subarea. When robot moves, the boundary is changed accordingly. However, there is no communication between robots. Robots not only update their locations to sensors inside their subarea, but also sensors outside. When sensors receive this kind of location message, they will check if the robot is nearer than that is assigned. If this is true, sensor changes its robot. In this way, the virtual boundary is modified. In sum, the dynamic and centralized algorithm has lower motion overhead. The fixed and dynamic algorithm has high message overhead. The centralized and dynamic algorithm can deal with unevenly sensor failure events better.

This assumes robot always has sensor in "hand", which fails to consider capacity of robot. Due to large scale of network, this is unrealistic.

### 2.3.2 Hexagon Approach

Falcon et al. presented a protocol named Carrier-Based Coverage Augmentation (CBCA) [8]. In this algorithm, robots have limited capacity, and they have to return the base station to refill sensors when all the sensors they carry is deployed. It focuses on the situation that some deployed sensors fail and recover the sensing hole. There are two parts in this algorithm, Reactive Advertising Routine (RAR) and Iterative Sensor Placement (ISP). In RAR, sensors transmit "hello" message periodically, and in this way, sensors could detect the failure of their neighbours if they cannot hear "hello" in a certain period of time. Consequently, they will report the failure to the border sensors, and the border sensors keep the record of sensors' failure.

Fig. 2.8 illustrates how RAR works. The sensor $V$ doesn't work and its neighbour $M$ detects this failure. $M$ sends a notification message greedily outward, from the inside hexagon to outside hexagon. The message node reach border node $S_1$, and propagates along the network border in Face Routing mode. It stops at border node $S_2$ because it has already received a notification from $F$. After receiving the notification, $S_2$ starts another round of border traversal by another failure notification. In this way, border nodes have a knowledge of failure information inside. For instance, $N$ knows there is a failure at $V$ which is provided by RAR.

In ISP, when a robot comes into the environment, it marches to the point of interest beaconing with its destination all the time. When a sensor receives this beaconing message, it replies "hello" message with failure information it keeps inside the network. The robot then stops marching and beaconing and enters a search phase. After it knows a location to deploy a sensor, it engages a migration phase in which it marches without beaconing to discover the vertex to lay a sensor. When the robot $B$ arrives to $V$, there may be some other robots approaching to $V$ as well or some robot has deployed a sensor at $V$ already. To avoid the multiple deployment, $B$ needs a

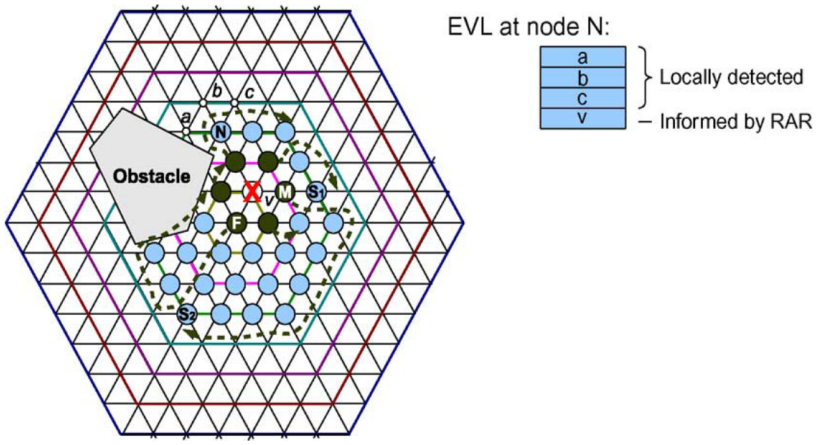



Figure 2.8: Reactive Advertising Routine

period of time to make sure this point is still void. After that, it deploy a sensor at $V$.

Fig. 2.9(a) illustrates a competition case between robot A and B, the one with smaller ID wins the competition. In the case of Fig. 2.9(a), $A$ wins and $B$ hear the "hello" message from the sensor at $F$ and knows there is a void vertex adjacent to $F$. $B$ marches to that place and deploy a sensor there. Fig. 2.9(b) illustrates that $B$ marching outside and hit the border of the network and get the information that there is a vertex available due to the failure of previous sensor occupying there. Then it marches to that point and make a deployment. Border of the network keeps record of information (e.g., failure nodes and also if the task of filling specific hole is taken by a robot).

As these two solutions didn't consider to use redundant sensor already on topology to fill sensing holes, they are dealing with coverage repair problem rather than sensor relocation problem.

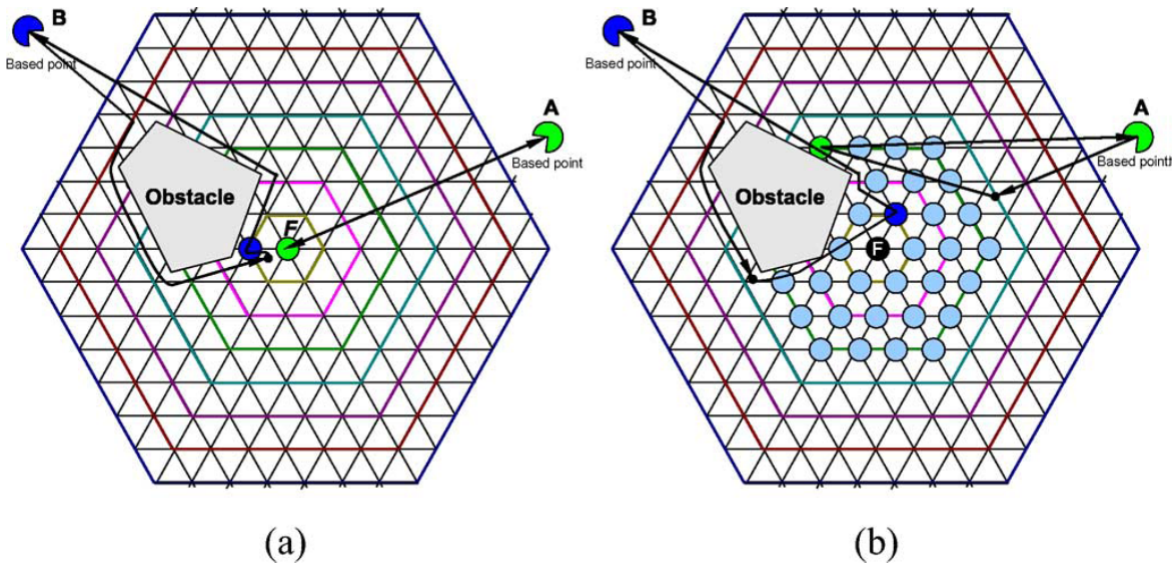



Figure 2.9: Iterative Sensor Placement (ISP) (a)Competition at F (b) Ordinary condition

## 2.4 Carrier-based Sensor Relocation

### 2.4.1 Ant Colony Approach

Falcon et al. introduce a novel combinational optimization problem: the one-commodity traveling salesman problem with selective pickup and delivery (1-TSP-SELPD) [9]. This algorithm is designed to deal with coverage repair problem in wireless sensor network. Robot move redundant sensors and deliver to fill sensing holes if any exist.

Ant colony optimization can solve the problem within reasonable time. The authors propose six ACO heuristic functions guide robots pick up and drop off behavior. There is a base station used to collect data. Redundant sensors and active sensor with a sensing hole nearby report to base station periodically. The problem is represented by a graph in which vertices are sensing holes or spare sensors and edges are transmission paths for ant. In this case, spare sensors are not necessarily to be visited,

while sensing holes have to. The ACO heuristic function is related to ants current cargo and distance between two nodes. For example, when the cargo of the ant is full, the ant should not fetch passive sensor any more. Instead, it should go to a sensing hole and deploy a sensor there.

### 2.4.2 Grid Approach

In [27], the whole ROI is divided into several grids. In each grid, a head sensor collects information from its grid members to determine sensing holes and redundant sensors. Sensing holes are regarded as a single point in this algorithm. Robot navigates in the targeted field and collects information from head sensors. In the process of collecting information, robot relocates redundant sensors at sensing holes within the same grid. Here robot is prone to take the sensor with much energy left and fill a nearest sensing hole. When exploring process ends, with the view of the whole field, this algorithm runs as a centralized one. Robot then collects redundant sensors in targeted area to fully fill its capacity. This spare sensor collection process is solved similar as Knapsack Problem (KP) to maximize the total energy of collected redundant sensors. Then they consider the hole filling process as Travelling Salesman Problem (TSP). A route for filling sensing holes is designed according to minimum spanning tree in which leaves are sensing holes. This algorithm is only applicable for single robot scenario. Besides, it is inefficient to collect redundant sensors until capacity is full without deploying sensors "at hand" even if robot happens to pass by some sensing holes.

### 2.4.3 Randomized Approach

Li et al. proposed a set of Randomized Robot-assisted Relocation of Static Sensors (R3S2)[15] algorithms for coverage repair and sensor relocation. Four algorithms, R3S2, G-R3S2, C-R3S2 and C-G-R3S2 are presented here. In these algorithms, robots

search in the environment collecting redundant sensors and deliver them to cover sensing holes. In R3S2, the robots make completely random movement in ROI. They fetch encountered spare sensors and deploy them at sensing holes. In G-R3S2, the robots' random movement is restricted on the virtual grids and robots move in the least visited direction so as to maximize the probability to discover more sensing holes and redundant sensors. C-R3S2 and C-G-R3S2 are cluster-based algorithm that cluster heads collect information of boundary nodes and optimize the deployed location.

In R3S2, the robots march in the environment without communicate with sensors. Between two moving sessions, there is a discovering phase in which the robots communicate with sensors to see whether there are passive sensors to pick up or sensing holes to repair. Fig. 2.10 indicates an example of local sensing hole identification. S has two uncovered arcs, so there are two sensing holes near S. There are four situations for the discovering phase. Let $\mathcal{H}$ and $\mathcal{V}$ denote the hole set and passive sensor set discovered by robot set $\mathcal{R}$ during discovering phase. If both $\mathcal{H}$ and $\mathcal{V}$ are empty, robot will continue to make random movement. If neither $\mathcal{H}$ nor $\mathcal{V}$ is empty, robot R will choose a redundant sensor to fill a sensing hole so that $|RV|+|HV|$ is minimized. If $\mathcal{V}$ is empty while H is not, R will choose a nearest hole to repair if it is carrying some sensors. If $\mathcal{H}$ is empty while $\mathcal{V}$ is not, R decides to pick a closest sensor up at probability p, where p = 1 if S is empty, and $\frac{1}{|S|}$otherwise.

If some robots are neighbors, they will compete for some common tasks. For example, if robots are competing to pick up spare sensors, the robot with less spare sensors wins. If two robots are competing for repairing a sensing hole, the robot with more spare sensors wins.

G-R3S2 imposes some constraints on robots movement to shorten the expected traversal time. In G-R3S2, the movement is restricted along grids. In addition,

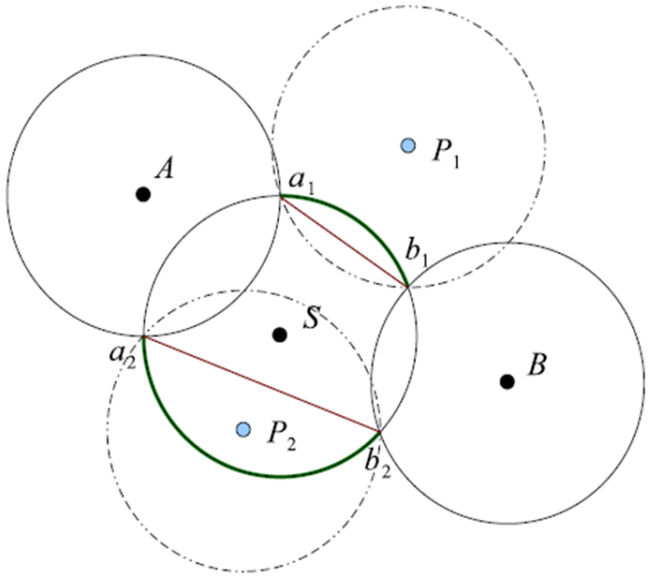



Figure 2.10: Local sensing hole identification

G-R3S2 further restricts the movement by using a Least Frequently Visited (LFV) policy [2]. Each grid selects the nearest sensor as "proxy". The proxy of grid records number of times that the grid is visited by each robot and instruct robot to move to the least recently visited neighbor grid.

In C-R3S2, Connected Dominating Set (CDS) is adopted to build cluster. The sensors in ROI are either in CDS or have a member in CDS. Sensors in CDS act as cluster heads and the remaining act as cluster member. With more than one cluster head available as one-hop neighbor, sensor chooses the cluster head with largest sensor ID as its cluster head. A node on the boundary of a hole collects the information of the hole and sends it to cluster head. Receiving coverage hole information, cluster head can merge the reported sensing hole if they are continuous. Then cluster head adopts a Virtual Force Approach (VFA) to find better deployed location. The initial deployed location choosing strategy is presented in R3S2. Cluster head set the distance between a boundary node $S$ and center point of the hole as nodal separation $d_{th}$. If the distance

between another sensor with the deployed location is longer than $d_{th}$, the sensor presents an attractive force. Otherwise, sensor presents repulsive force. When the whole process ends, the new sensor is virtually moved to a place so that the sensing overlap is reduced. Although this kind of deployment strategy is better compared to the one in R3S2, it may still generate small new holes. In addition, cluster-based approach requires the topology of sensors to be connected. In our case, assuming sensors randomly scattered in ROI, connected graph is not guaranteed. Due to the limitation of cluster-based approach, we only consider R3S2 and G-R3S2 as competing algorithm. The performance of G-R3S2 is better than R3S2 on all metrics. The algorithms terminate either all redundant sensors are used or there is no more coverage holes. The deployed location choosing strategy fails to take progress of a new sensor into consideration. Therefore, robot always chooses to fill the nearest hole even if the hole is small. This deployed location choosing strategy causes limited performance for one iteration of relocation.

## 2.5 Mobile Robot Task Allocation

### 2.5.1 Tree-based Approach

Coltin et al. present three algorithms to assign mobile sensors to events [7] detected by static sensors in hybrid wireless sensor network, which are centralized algorithm, auction-based algorithm and tree-based greedy algorithm separately. In these algorithms, static sensors are responsible to detect events must be handled by robots. The optimization goals are Minimizing distance travelled by mobile nodes and communication among static sensors. In centralized algorithm, a leader is responsible to assign robots to events. The leader always greedily selects robot nearest to a specific event. In this case, fewer messages are needed. The second one is a distributed

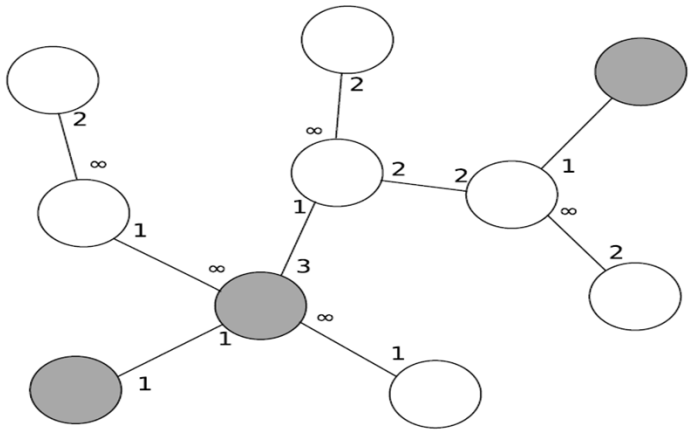



Figure 2.11: A tree indicating distance to nearest node which needs mobile robots

auction based algorithm. Mobile sensors without tasks announce an auction to the nearest static sensor and the static sensor percolates the announcement to the entire network. Upon receiving such announcement message, static sensor would create a list for the message and sends a bid to the nearest robot. Robots may receive more than one bid, and it always chooses the best one which is sent by the nearest static sensor. The third one is a tree based greedy algorithm. All static sensors greedily form a minimum spanning tree. The mobile robot without task sends its location to the nearest static sensor. Unlike auction-based algorithm, which is reactive in the sense that static sensors only respond to mobile robots auction, static sensor in tree-based algorithm proactively form a model of where mobile robots are needed before any requests arrive. Rather than modeling the number of mobile robots at every static sensor, this algorithm only store, for each sensor, the distance to the nearest static sensor which needs mobile robots from each outgoing edge. The mobile robots available sends the request to the nearest static sensor, if this sensor needs robot, it takes this robot. Otherwise, the sensor forwards the request to the closest node in the tree which needs mobile robots. In this algorithm, the request does not traverse

through the whole network, but to the nearest node with need on the tree. As this is a distributed algorithm, its more robust compared to a centralized algorithm.

Fig. 2.11 shows an example of the tree we just referred to. Grey nodes in this figure need mobile robots. Numbers indicate the distance to nearest node which needs mobile robot. $\infty$ indicates there is no node that needs mobile robot in this direction.

### 2.5.2 Auction-based Approach

Mezei et al. present Simple Auction Protocol (SAP), Improved SAP and Aggregation Protocols [23]. It is a localized algorithm for actor-actor coordination based on auction protocol. The request for service is collected by actuators and each actuator replies an offer with cost to do this work. This algorithm can find a closest actuator because all actuators are consulted. However, sometimes even the best actuator is near the event, the response time can be large because the actuators have to wait until all of them receive request and choose a best one. The authors propose k-SAP and k-SAAP to improve algorithms in Communication aspects of Coordination in Robot Wireless Networks. The first one is an improvement for SAP. Instead of flooding in the whole network, actuators only search in their k-hop neighbors. For example, for A1, 1-SAP means A1 only collects bids from A2 and A3. 2-SAP floods to A2, A3, A4, A5 and A8. 3-SAP consults all nodes. Another type of improvement for SAP is to reduce message overhead. Instead of using separate routes to relay messages, the new algorithm uses trees to report back. As it is shown in Fig. 2.12, tree expansion starts from A1 which serves as tree root. A2 and A3 retransmits search message. From the figure we can see that A1 is A2 and A3's parent. One node can only have one parent. For example, only A4 is A9's parent even though A5 and A9 are also connected. Leaves report their bids to parent, and parent compares the bids. Then parent node only reports the best bid it receives to its parent. In this way, message overhead is reduced. To

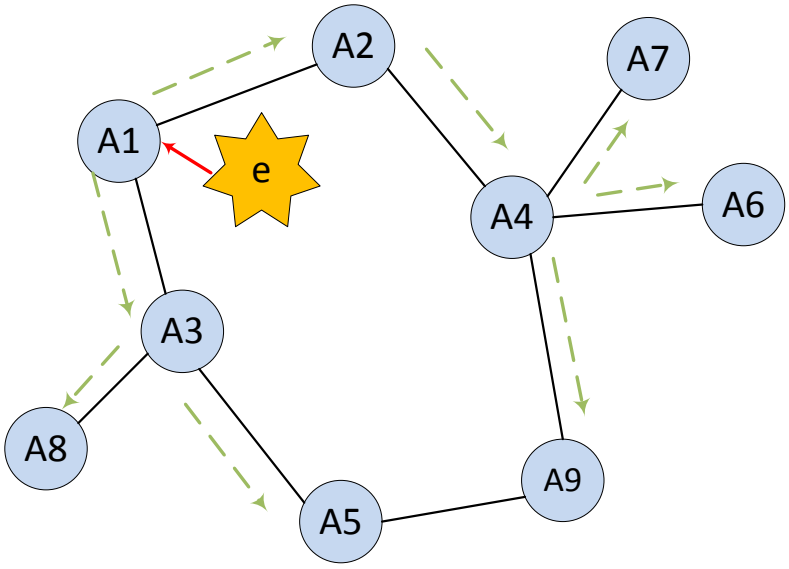



Figure 2.12: Auction aggregation protocols

enhance k-SAP, the authors propose a greedy improvement (k-SAPG). After k-SAP finished, one best actuator is decided. At this time, one hop SAP can be initialized to see that whether best actuator's one hop neighbour is better. The algorithm repeats until no better actuator is found.

## 2.6 Sensor Active Scheduling

Gallais et al. present several localized sensor area coverage protocols [12], for arbitrary ratio of sensing and transmission radii. Sensors are assumed to be synchronized and sensors are judged whether to be active at the first time. This approach has a very small communication overhead since prior knowledge of neighbors is not required. Neighbor information is brought by activity messages. Authors present a geometric theorem, which is applicable for any ratio of sensing and communication radii. Moreover, it is applicable to any shape of monitored region by a sensor. A

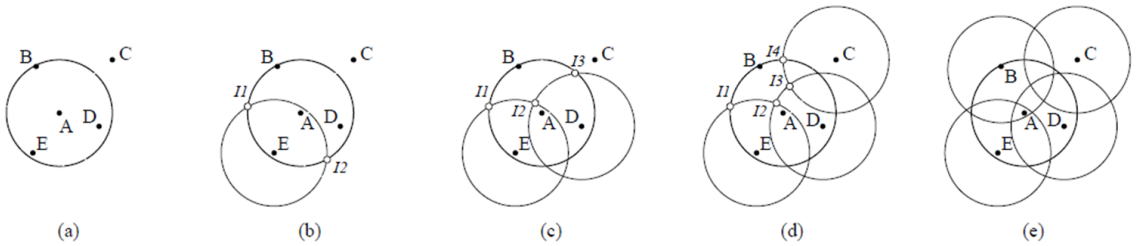



Figure 2.13: Intersection-based coverage evaluation scheme

theorem efficiently confirms whether or not a sensing region is fully covered by other regions is presented: If there are at least two covering circles and any intersection point of two covering circles inside the sensing area is covered by a third covering circle, then the sensing circle is fully covered. We can see the judging process of whether a sensors covering area can be fully covered by its neighbor is shown in Fig. 2.13. The circle in Fig. 2.13(a) indicates $A$s sensing circle. Sensing circle of neighbor $E$ has two intersections, $I_1$ and $I_2$ with $A$. When covering circle of $D$ is involved, $I_2$ in Fig. 2.13(b) is covered and 2 other intersections are generated as it is shown in Fig. 2.13(c). When $B$ and $C$ are taken into account, all intersections on the boundary or inside of $A$ are covered. According to the theorem, $A$s covering area is fully covered by its neighbors.

# Chapter 3

# Algorithm MSR

As discussed in Chapter 1, no satisfactory carrier-based localized sensor relocation exists. The only existing solution for carrier-based sensor relocation relocation are R3S2 and its improved version G-R3S2 [15]. The relocation delay of these approaches is long, and they are not energy saving for robot.

Market-based Sensor Relocation (MSR) is a localized carrier-based sensor relocation algorithm. We will discuss it in details in this chapter.

## 3.1 Model and Definition

We assume identical sensors randomly scattered in ROI. Let $\mathcal{S} = \{S_0, S_1, ..., S_{n-1}\}$ be the set of $n$ sensors, and $\mathcal{R} = \{R_0, R_1, ..., R_{m-1}\}$ the set of $m$ robots for replacing sensors. As we referred to above, the continuous uncovered area is called *sensing hole*. Some sensing holes near the edge of ROI may not be enclosed by sensors. The boundary of this kind of sensing hole is likely to be large, which causes the inefficiency to identify the sensing hole. As the boundary of the ROI is predetermined, one can pre-deploy a set of connected sensors on the boundary of ROI, so that each sensing hole is fully surrounded by sensors. Further, some big sensing holes could be similarly

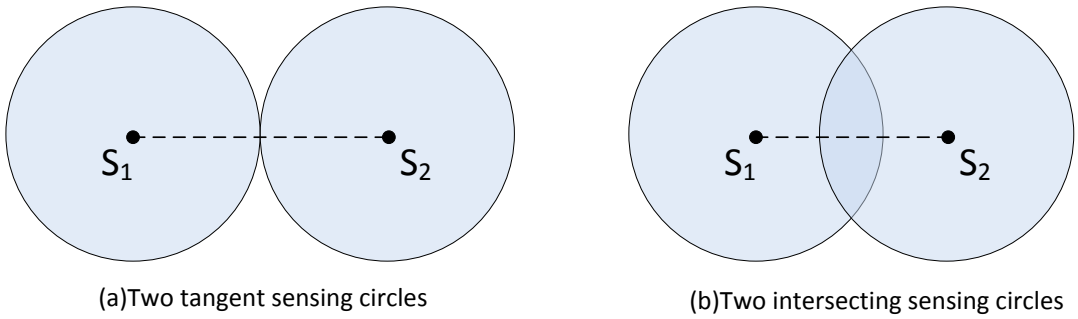



Figure 3.1: Relationship between $r_s$ and $r_c$

divided into small ones. In our performance analysis, we will consider both scenarios, regular ROI and ROI with enclosed sensors, in Chapter 4.

Spare sensors "sleep" to save energy. Non-redundant sensors must remain *active* for monitoring surrounding area. Sensors within *communication range* $r_c$ can communicate with each other. Each sensor can provide circle-shaped coverage within a radius of *sensing range* $r_s$. We assume $r_c \geq 2r_s$ to guarantee any two sensors having coverage overlap are connected. As it is shown in Fig. 3.1(a), since the two sensing circles are tangent, the distance between $S_1$ and $S_2$ is $2r_s$. In our algorithms, we need two sensors with intersecting sensing circles talk with each other, which is the case illustrated in 3.1(b). As the distance $d(S_1, S_2) < 2r_s$, these two sensors can communicate with our assumption $r_c \geq 2r_s$.

Robots are used to fetch spare sensors in ROI and relocate them at sensing holes. Robots have unique IDs and same communication range $R_c$. LRV strategy[2] is adopted for robot to explore ROI. Robot can carry only one sensor. Sensors and robots are aware of their locations with help of GPS.

An activity scheduling protocol [12, 31] is adopted for determining spare sensors. Sensors locally identify their status using only information from their one-hop neighbors. Such algorithm, based on local knowledge, is called localized. Our goal is to

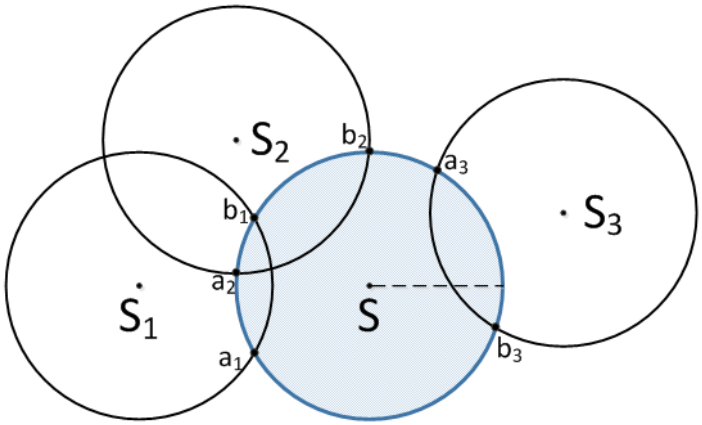



Figure 3.2: Uncovered arc detection

design an efficient localized algorithm for robots to collect passive sensors and deliver them to sensing holes. As we discussed in Chapter 2, an activity scheduling protocol [12] is adopted for determining spare sensors. Sensors locally identify their status with only information of their one-hop neighbors. Our goal is to design an effective and efficient algorithm to use a team of robots to collect spare sensors and fill sensing holes with them.

## 3.2 Sensing Hole Identification

### 3.2.1 Uncovered Arc Detection

We choose a representative for each sensing hole to collect the information of the hole and send service request to robot.As we discussed in Chapter 1, active sensors are responsible for providing coverage. They periodically sends "Hello" message containing their location information to inform their existence to neighbors. Active sensor learns the presence of active neighbors via "Hello" message. As it is shown in Fig. 3.2, $S$ knows the locations of three neighbors $S_1$, $S_2$ and $S_3$ respectively. According

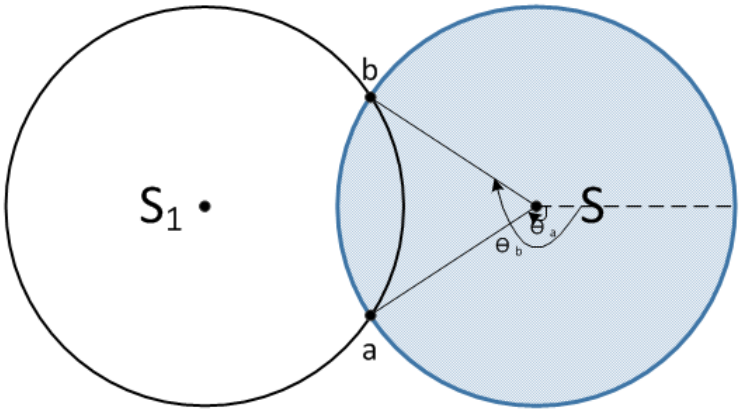



Figure 3.3: Using radian to represent arc of a node

to the three neighbors' and also its own location, through simple calculation, $S$ gets intersections with them, which are $a_1$, $b_1$, $a_2$, $b_2$, $a_3$ and $b_3$. From Fig. 3.2, we can see that $\widehat{a_1b_1}$, $\widehat{a_2b_2}$, $\widehat{a_3b_3}$ are covered by $S_1$, $S_2$ and $S_3$ respectively. As it is shown in Fig. 3.3, we use radian to represent arc. We define east and south to be positive direction of axis X and Y. So $\widehat{ab}$ can be illustrated as $[\theta_a, \theta_b]$. In Fig. 3.2, $\widehat{a_1b_1}$ covered by $S_1$ leaves uncovered arc of $S$ to be $[0, \theta_{a_1}] \cup [\theta_{b_1}, 2\pi)$. Considering $S_2$, since $a_2 < b_1$, $\widehat{b_1b_2}$ is additionally covered by $S_2$. Then uncovered arc turns into $[0, \theta_{a_1}] \cup [\theta_{b_2}, 2\pi)$. Finally, $\widehat{a_3b_3}$ is covered. In the condition that all circles are in same size, one can only cover arc of another circle with less than $\pi$. In this case, $a_3 > b_3$. If $\theta a_3 - \theta b_3 > \pi$, the covered arc expressed by radian is $[0, \theta b_3] \cup [\theta a_3, 2\pi)$. Then, $[\theta b_3, \theta a_1] \cup [\theta b_2, \theta a_3]$ is left. Thus, there are two uncovered arcs for $S$, $\widehat{a_1b_3}$ and $\widehat{a_3b_2}$. Fig. 3.4 shows an example that the sensing circle of $S$ is fully covered by $S_1 \sim S_5$.

### 3.2.2 Choosing Representative

As we referred above, through local calculation sensor determines whether its sensing circle is fully covered by its one hop neighbors. If it is not, there is at least one sensing hole nearby. This kind of node is defined as boundary node. Representative

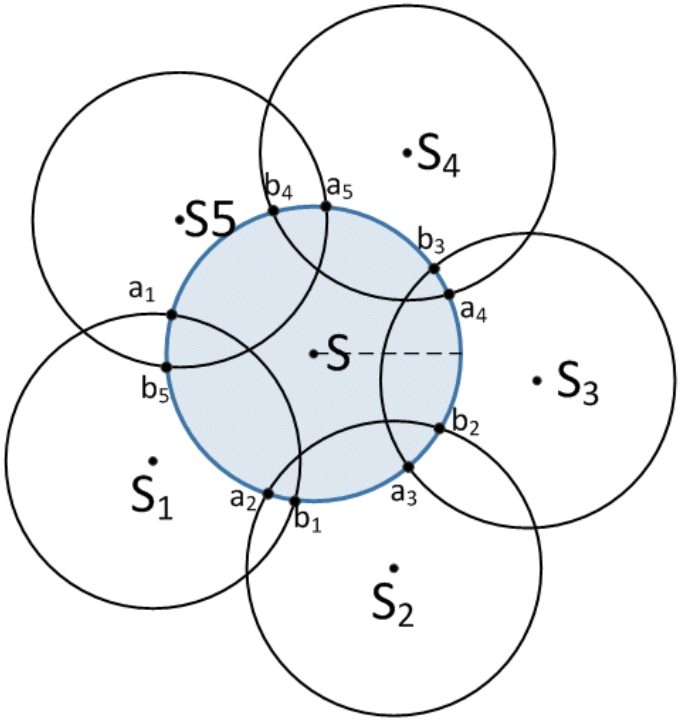



Figure 3.4: Sensing circle fully covered

is chosen to keep record of information of the whole boundary. As boundary nodes don't have knowledge of the whole coverage hole at the first time, the approach to choose representative from candidates is as follows:

The boundary nodes initiate "boundary search" message travelling in counter-clockwise manner. As sensing hole is enclosed by uncovered arcs of boundary nodes, "boundary search" messages can travel along the boundary along uncovered arcs. Each boundary node records whether there are such messages relayed to other nodes through specific uncovered arcs. In addition, initiator's ID (*largestInitID*) of messages relayed along specific arc is also stored in boundary nodes. When a boundary node notices the coming of a "boundary search" message with a initiator's ID (*msgInitID*), it relays the message only if this arc has not relayed any messages or *msgInitID* is larger than *largestInitID*. Once the message relayed, the value of *largestInitID* is

updated. Also, information about this boundary node (*sensor ID*, *location*) is inserted in the relayed message. Otherwise, the message is not relayed, which means traversal of this message is blocked. Algorithm 1 presents a the rule to determine whether to relay "boundary search" message. In this way, only the message initiated by the candidates with largest ID is able to traverse the whole boundary and return to the initiator. This is elected as boundary representative. Representative has knowledge of all the boundary nodes.

**Algorithm 1** Determine whether to relay messages

```
Initialization:
  if S is candidate then
    largestInitID ← sensorID
  else
    largestInitID ← 0
  end if
  relay ← false
Message Coming:
  if LargestInitID = 0 then
    relay ← true
    largestInitID ← msgInitID
  else
    if msgInitID > largestInitID then
      relay ← true
      largestInitID ← msgInitID
    end if
  end if
```

Fig. 3.5 provides an example for hole identification. There are three sensing holes in this figure. The sensing hole on left and right side are not shown completely. Due to the set of connected sensor pre-deployed on the boundary of ROI, these two holes can also be enclosed by some sensors as the one in the middle. Here we consider the middle one colored pink which is surrounded by $S_2 \sim S_7$.

Boundary nodes initiate "boundary search" message through their uncovered arcs $\widehat{AB}, \widehat{BC}, \widehat{CD}, \widehat{DE}, \widehat{EF}$ and $\widehat{FA}$. First of all, we consider the "boundary search"

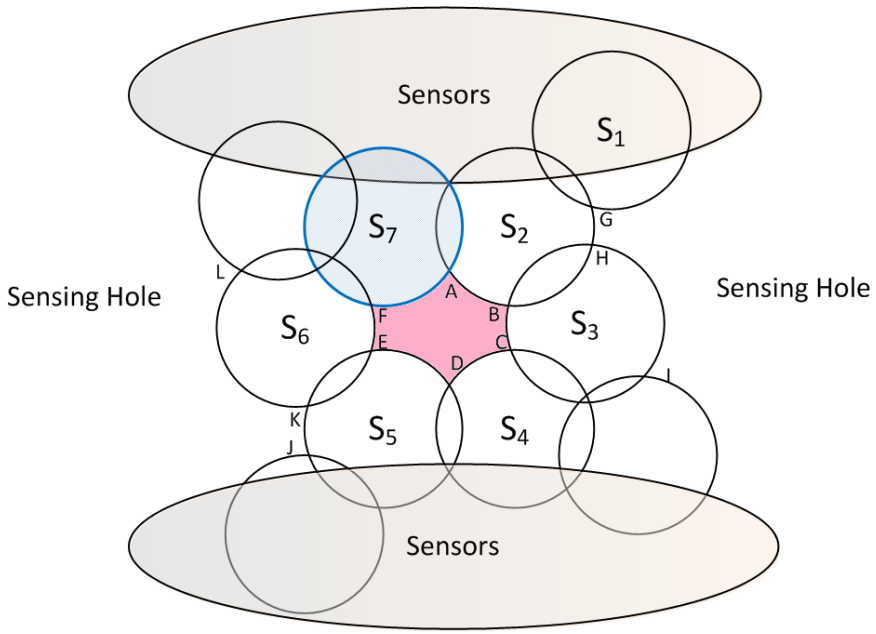



Figure 3.5: An example for sensing hole identification. $S_7$ is chosen as representative.

message initiated by $S_4$. When $S_3$ notices the coming of "boundary search" message from $S_4$, $S_3$ decides to relay it since *msgInitID* 4, is larger than *largestInitID* 3. When $S_7$ notices the arrival of this message, as *largetInitID* of $\widehat{AF}$ 7 is larger than *msgInitID* 4, this "search message" is blocked at $S_7$. In this way, only the message started by $S_7$ is able to traverse the whole boundary and return to itself which can be selected as the representative of the coverage hole.

## 3.3 Deployment location choosing

As the representative has obtained locations of all the boundary nodes, it can find several "best" locations to cover detected sensing hole. To reduce the complexity of calculating area of polygon-shaped sensing hole, it can be assumed to grow proportionally to $k-2$, where $k$ is the number of arcs on the boundary. In this analogy, to reduce the size of sensing hole, we need to decrease the number of arcs and intersections on the boundary, by attempting to cover three consecutive intersections

with a relocated sensor. The sensor generates a new arc while covering two existing ones on the boundary, which decreases total number of boundary arcs by one. A new sensor covering more intersections is less likely to generate small holes. Due to the limited size of sensing circle, a sensor can rarely cover four or more intersections. Consequently, we look for locations (for deploying a sensor) to cover three intersections. We further improve the method by maximizing additional covered area to find a "best" possible location. We now present details of this method.

Robot first searches for a location $D$ where deployed sensor is able to cover three consecutive intersections. The longest edge connecting three intersections is defined as $L$, with length $|L|$. The opposite angle of $L$ is $\alpha$. Tie is resolved randomly.

**Algorithm 2** Choosing a location covering three intersections

**Input:** $L_M$, $\alpha_M$
**Output:** *DeployLoc*
  **if** $L_M > 2r_c$ **then**
    $DeployLoc \leftarrow null$
  **else if** $L_M = 2r_c$ **then**
    **if** $\alpha < \pi/2$ **then**
      $DeployLoc \leftarrow null$
    **else**
      $DeployLoc \leftarrow$ Midpoint of the edge with length of $L_M$
    **end if**
  **else**
    **if** $\alpha_M \geq \pi/2$ **then**
      $DeployLoc \leftarrow$ Midpoint of the edge with length of $L_M$
    **else if** Radius of circumcircle of the triangle $r \leq r_s$ **then**
      $DeployLoc \leftarrow$ Circumcircle center
    **else**
      $DeployLoc \leftarrow null$
    **end if**
  **end if**

In Algorithm 2, the term "null" means that the representative cannot find location $D$ that can cover three intersections. Solutions for this case will be discussed later. We first discuss how to further improve this initial location for $D$ (when the algorithm

finds it), leading to reduction in hole size. The representative adopts a *greedy virtual move policy* to further optimize deployment location. Let $\mathcal{D} = \{Dn, Ds, Dw, De\}$ denote the set of locations in north, south, west and east direction of $D$ with distance of fixed predetermined step length. The additional area covered by newly deployed sensor at a point $D_i$ is denoted by $A(D_i)$, where $A(D_i)$ is the difference between the total sensing area and overlapped area. For $\forall D_i \in \mathcal{D}$, the representative $S_r$ checks if sensor at $D_i$ could still cover three consecutive intersections. Every location satisfying this restriction is included into a set $\mathcal{D}' \subseteq \mathcal{D}$. Furthermore, we define $Dmax$ as the point $D_i$ from $\mathcal{D}'$ that maximizes $A(D_i)$. $S_r$ virtually moves the sensor to $Dmax$ if $A(Dmax) > A(D)$, and updates $D$ as $Dmax$. Information in sets $\mathcal{D}$ and $\mathcal{D}'$ is also updated in turn. The process iteratively repeats at new $D$ until $\max_{D_i \in \mathcal{D}'} A(D_i) \leq A(D)$. After this iteration terminates, the current $D$ is defined as $D'$ which is a locally optimal best deployment location.

Fig. 3.6 shows part of boundary of a sensing hole. Considering three intersections $A$, $B$ and $C$ in Fig. 3.6(a), the longest edge $L$ connecting these three intersections is $AC$ and the opposite angle $\alpha$ is $\angle ABC$. Further, $|L| < 2rs$ and $\alpha > \pi/2$. Thus $D$ locates at the midpoint of edge $AC$ according to Algorithm 2. The new sensor is virtually moved according to the policy mentioned above. The arrow in Fig. 3.6(b) shows the moving path of the sensor. When the sensor is moved to $D'$ indicated in this figure, $\mathcal{D}'$ only contains $De$, and $A(De) < A(D)$. Therefore the sensor cannot be further moved to increase additional covered area with still covering three intersections. Thus $D'$ is the final location for the new sensor.

Then we compares the difference of deployed location choosing strategy of the MSR and G-R3S2. As it is shown in Fig. 3.7, robots in two algorithms consider to deploy a sensor to cover the uncovered arc *wideparenab* of a boundary node $S$. In Fig. 3.7(a), intersections $a$ and $b$ are right on sensing circumference of new sensor.

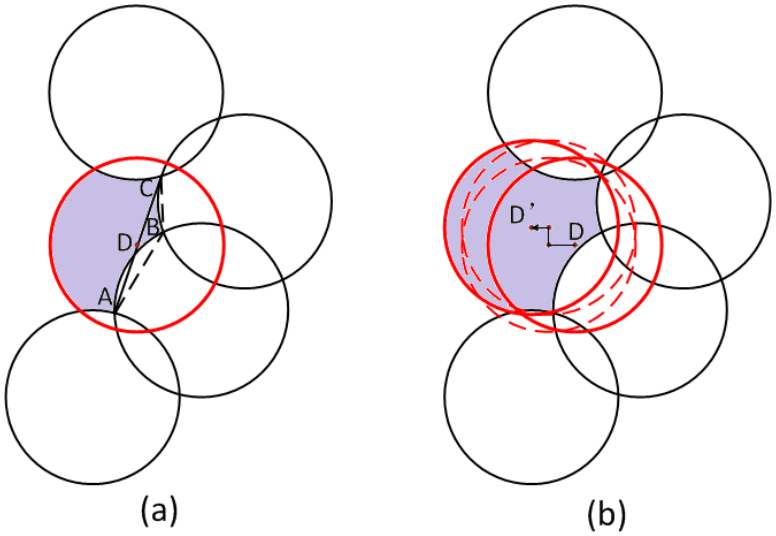



Figure 3.6: Sensor located at $D$ covers three intersections $A$, $B$, and $C$. The location improves after 4 iterations to $D'$ and cannot improve further.

However, this new sensor generates small sensing holes colored grey which requires an extra redundant sensor to cover. In Fig. 3.7(b), the new sensing circle covers $a$, $b$ and $c$ without generating small sensing hole. Although new sensor covers larger uncovered area, covering only two intersections is likely to generate small holes. This requires more spare sensors to cover existing sensing holes.

We are now back to resolve the case when the representative $S_r$ cannot find any location for virtual sensor that can cover at least three consecutive intersections within the sensing hole. $S_r$ then lowers selection requirement by searching location to cover only two intersections, to fully cover an uncovered arc. If $UV$ is one such arc then the new sensor is deployed at one of intersections of circles centered at $U$ and $V$, with radius $r_s$. $U$ and $V$ are then located on the sensing perimeter of new spare sensor. Consider an example in Fig. 3.8. Three deployed sensors create a large unbounded hole, and $UV$ and $XZ$ are two arcs on the boundary. Circles centered at $U$ and $V$ with radius $rs$ intersect at $D_0$ and $D_1'$. Newly deployed sensor centered at $D_1'$ (shown as shaded) has $U$ and $V$ on its perimeter. Similarly, a new sensor deployed at $D_2'$

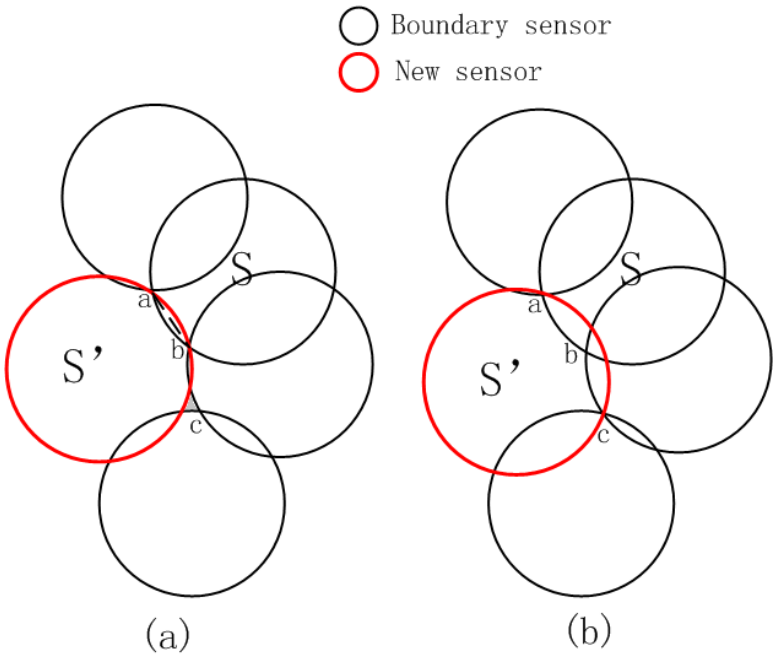



Figure 3.7: Comparison of deployed location. (a) is the deployed location of G-R3S2 and (b) is the deploy strategy of MSR

covers $X$ and $Z$.

## 3.4 Bidding Process

The representative sensor may receive from other sensors on the same hole, or calculate itself, several possible deployed locations $D'$ and additional coverage areas gained $A(D')$ at location $D'$. As we discussed in Section 3.3, the new sensor covering three consecutive intersections is less likely to generate small sensing holes, so we increase the weight of three intersection covering situation. Progress made by relocating a spare sensor is measured according to the following policies:

1. In the case of covering two intersections,

   $progress(D') = 2 \times A(D')$

2. In the case of covering three intersections,

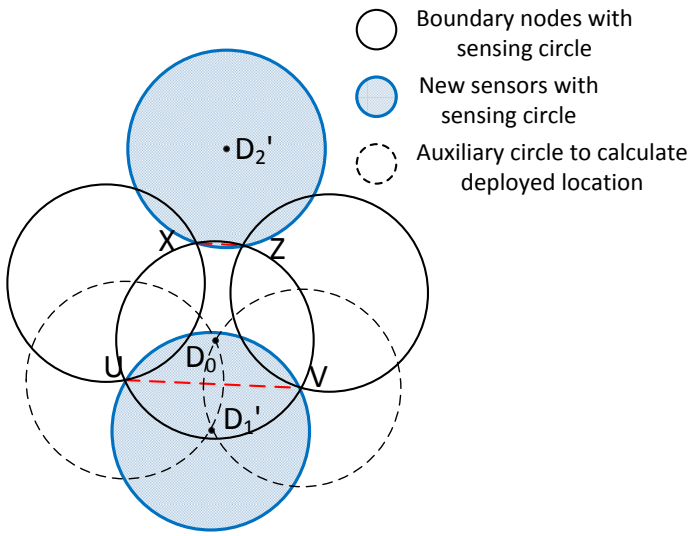



Figure 3.8: Deployed location choosing strategy for covering two intersections

$$progress(D') = 3 \times A(D')$$

The hole representative shares information about possible deployment locations with other boundary sensors along the same hole. As we referred in Section 3.5, $R$ moves in LRV manner. A virtual grid is added in ROI and $R$ moves along the edge of the virtual grid to explore the ROI. When $R$ reaches a grid point, it calls for information of spare sensors and sensing holes in the neighborhood by broadcasting "information collection" message. Receiving this message, boundary sensors within communication range bid to $R$ with possible deployed locations $D'$ and corresponding progress $progress(D')$. Besides, spare sensors also inform their existence to robot in vicinity. Let $\mathcal{P}$ and $\mathcal{V}$ be set of possible deployment locations and set of spare sensors discovered by robot $R$. When $R$ at grid point notices existence of a spare sensor $V$ or receives bidding message containing possible deployment location $D'$ from a boundary node, $R$ adds them into $\mathcal{V}$ and $\mathcal{P}$ respectively. *Cost* is defined as robot's travelling

distance to fetch a spare sensor and deploy it at a certain place, see Eq. 3.1.

$$Cost(V, D') = |RV| + |VD'|. \tag{3.1}$$

To find a best pair of $D'$ and $V$, we define the Cost over Progress ratio ($CoP$) in Eq. 3.2.

$$CoP(V, D') = Cost(V, D')/Progress(D'). \tag{3.2}$$

Robot compares *CoP(V,D')* value for each pair of $D'$ and $V$ and chooses the task with the minimal *CoP(V,D')*.

Here is the policy of robot's one iteration:

When a robot reaches a grid point, it gets $\mathcal{P}$ and $\mathcal{V}$ by collecting information of spare sensor and also possible deployed locations with corresponding progresses.

If the robot is not loaded with sensor, it makes decision as follows.

1. If $\mathcal{P} \neq \varnothing, \mathcal{V} \neq \varnothing$, robot takes the spare sensor $V \in \mathcal{V}$ and deploys it at $D' \in \mathcal{P}$ so that *CoP(V,D')* is minimized. Then robot goes to the nearest grid point.

2. If $\mathcal{P} = \varnothing, \mathcal{V} \neq \varnothing$, robot takes the nearest spare sensor and marches to the nearest grid point.

3. If $\mathcal{V} = \varnothing$, robot moves to a neighbouring grid point based on LRV policy.

If the robot is carrying a sensor, it makes decision as follows.

1. If $\mathcal{P} \neq \varnothing$, robot deploys the loaded sensor at $D' \in \mathcal{P}$ so that *CoP(V,D')* is minimized and marches to the nearest grid point.

2. If $\mathcal{P} = \varnothing$, robot moves to a neighbouring grid point based on LRV policy.

A new iteration begins once a robot arrives at a grid point. In one iteration, if robot has task to deal with, such as taking spare sensor and deploying sensor at

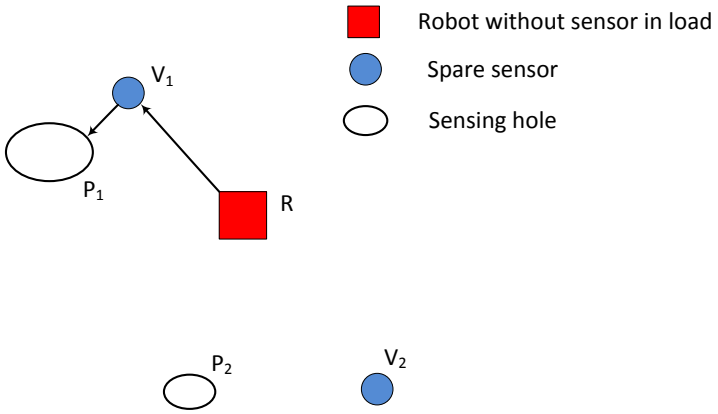



Figure 3.9: An example of robot without load making choice when there is multiple $V$ and $P$ available

specified location, the robot may need to move off the grid. After finishing a task, robot returns to the nearest grid point to start a new iteration. After a new sensor is deployed, it is selected as representative of the new boundary hole if sensing hole is not fully covered. Boundary nodes relay the "boundary search" message initiated by the new sensor and locally identify the new boundary. Representative of the new sensing hole obtains possible deployed locations and corresponding progress and shares the information with boundary nodes along the same hole. Boundary nodes then bids to robot once they receive "information collection" message. The procedure of relocation terminates when there is no more sensing holes or spare sensors, which is the same as the existing algorithm [15].

As it is illustrated in Fig. 3.9, robot without load receives information of two spare sensors and two possible deployed locations. There are two elements in both $\mathcal{P}$ and $\mathcal{V}$. In this figure, the size of oval represents the progress it would made to deploy a sensor at specified location. Of all possible pairs of $P$ and $V$, $P_1$ and $V_1$ provide smallest $CoP$ for $R$. So $R$ would pick up $V_1$ and relocate it at $P_1$.

Fig. 3.10 describes an example of robot's choice when it has sensor in load. In G-R3S2, robot would decide to fill the hole at $P_3$ since it is the nearest one. However,

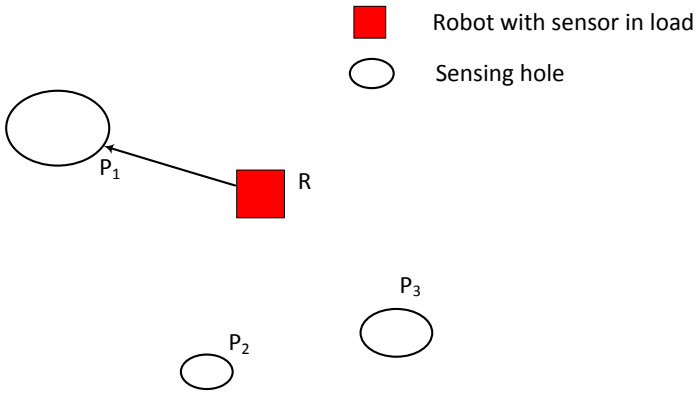



Figure 3.10: An example of robot with load making choice when there is multiple $V$ and $P$ available

even if the distance $d(R, P_1)$ is longer than $d(R, P_3)$, sensor deployed at $P_1$ would gain much more progress. According to the rule we referred to above, $R$ chooses to fill the hole at $P_1$ since less $CoP$ is achieved.

## 3.5 Moving Strategy

In MSR, robot moves randomly in ROI and relocates encountered spare sensor to sensing hole. In order to reduce the time to explore the whole ROI, some restrictions are brought in the moving strategy. The movement of a robot is restricted in a virtual grid when robot is in searching phase, which helps robot to explore the ROI in shorter time.

As we described in Section 3.4, when robot reaches to a grid point, it collects information of spare sensor and possible deployed location in neighborhood. We need to ensure all sensors in ROI have a chance to communicate with robot when robot is in exploring phase (collecting information at grid points). Therefore, some restrictions about the relation of length of square grid edge and sensor's communication range $r_c$ needs to be defined. Denote the length of the square grid edge is $L_G$. We require $L_G \leq \sqrt{2}r_c$.

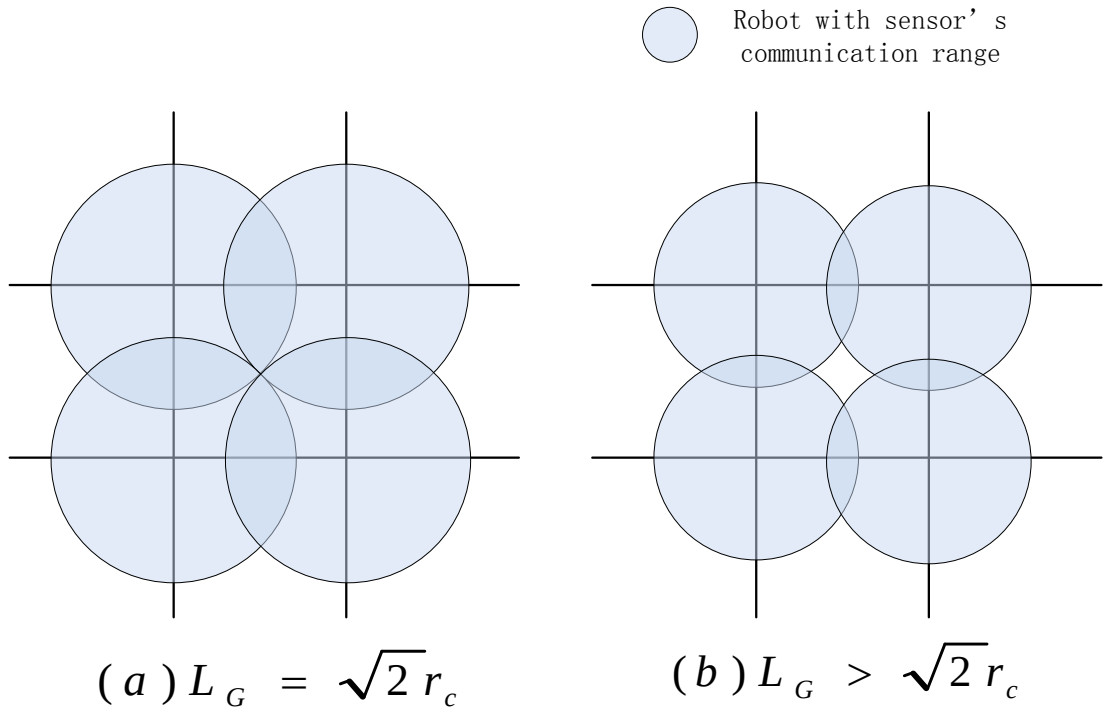



Figure 3.11: Relation between $L_G$ and $r_c$

Fig. 3.11 shows the relation between $L_G$ and $r_c$. The radius of the circle in the figure indicates length of $r_c$. Robots may stop at the four grid points in this figure to collect information. Robot at grid point generates a communicating circle with radius $r_c$, in which sensors can talk with the robot. We should guarantee that sensor located in this grid can be covered by at least one communicating circle. In Fig. 3.11(a), with the condition of $L_G = \sqrt{2}r_c$, the grid is fully covered by robot's communicating circle. However, in Fig. 3.11(b) where $L_G > \sqrt{2}r_c$, area not covered by any communicating circles exists. So the restriction of $L_G \leq \sqrt{2}r_c$ is required to ensure all sensors in ROI can report their information to robot.

In addition to restricting robot's movement along grids, a Least Recently Visited (LRV) policy [2] is applied. Each grid point chooses the nearest sensor as its proxy. Proxy keeps record of how many times robot visits this grid and instructs robot to

move to a least recently visited neighbor grid. Algorithm 3 illustrates more details about the mechanism of updating weights of different directions and instruction made for robot.

**Algorithm 3** Time of arrival counting for grid

**Input:** Direction of robot reaching the grid $D_r$ and weights of different directions $Weight$

**Output:** Direction of robot leaving the grid $D_l$

$Weight_{D_r} \leftarrow Weight_{D_r} + 1$
**if** only one element in $\min Weight$ which direction is $D_{min}$ **then**
  $D_l \leftarrow D_{min}$
**else**
  randomly choose a direction from $\min Weight$, $D_{rmin}$
  $D_l \leftarrow D_{rmin}$
**end if**
$Weight_{D_l} \leftarrow Weight_{D_l} + 1$

According to LRV policy, robot is instructed to move to least visited grid, which helps robot to traverse ROI in short time. Since least visited grids are more likely to have tasks remaining, robot's tending to move to these grids helps to reduce the total relocation time and saves robot's energy. Robot stops at grid point, collects information and deals with selected task according to Section 3.4. Then robot moves to the nearest grid point and collects task request again. If no task is available, robot explores ROI based on LRV policy.

## 3.6 Robot Cooperation

As robots travel in ROI, it is possible for two or more of them to be within their communication range. Further improvement may be made by cooperation among robots. Each robot sends messages to its neighboring robots asking for their bids and spare sensors. Neighboring robots respond with requested information. Received bids and robot's own bids are sorted by $CoP$ values. Then each robot only takes the best

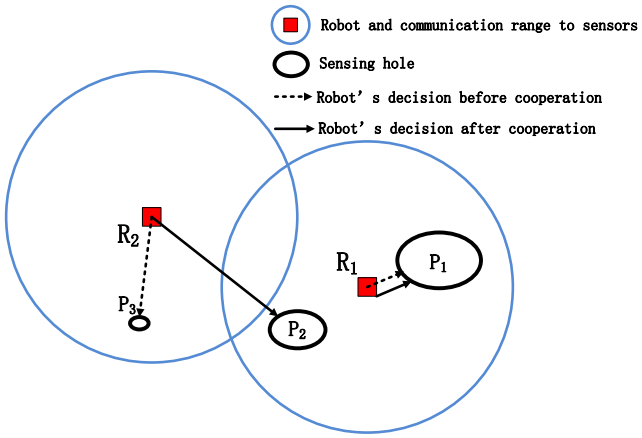



Figure 3.12: Robot $R_2$ learns about a large hole $P_2$ from robot $R_1$ and takes it instead of filling smaller but closer hole $P_3$.

bid with minimal $CoP$ and rejects the other bids. If robots' best bids correspond to the same hole or spare sensor, the robot with better $CoP$ wins. Other robots may then revise the selection of their best bid and try again to 'negotiate' with other robots in their vicinity. Further, robots may discover new neighbors. The communication and bidding may be then revised.

Robot cooperation provides more choices of spare sensors and sensing holes for robots. Fig. 3.12 shows an example of robot cooperation. The size of ovals in the figure indicates progress for each hole. Assuming both robots $R_1$ and $R_2$ have loaded sensor, according to their communication range, $P_1$ and $P_2$ is visible to $R_1$ and $P_3$ is available for $R_2$. $R_1$ and $R_2$ intended to fill the holes at $P_1$ and $P_3$ respectively. During communicating, $R_2$ leans the information of $P_1$ and $P_2$. $P_1$ is assigned to $R_1$ since $R_1$ can fill the hole $P_1$ with smaller $CoP$. For the two holes available for $R_2$, although $P_2$ is farther than $P_3$, the size of the hole at $P_2$ is much larger. This $R_2$ fills the hole at $P_2$ to minimize $CoP$. Robot cooperation may also lead toward exchanging their tasks.

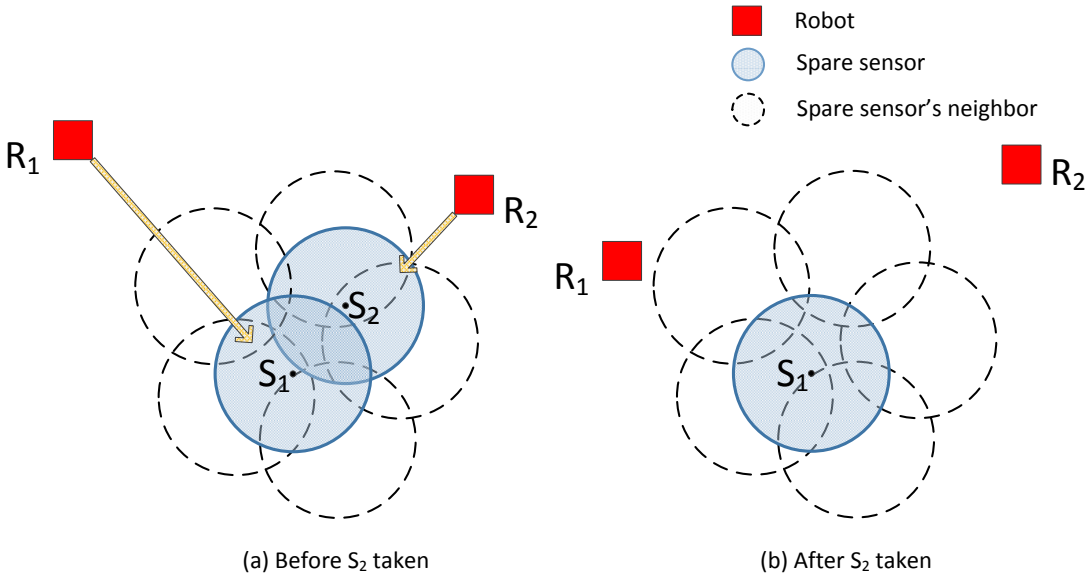



Figure 3.13: Removal of a spare sensor turns a neighbor spare sensor to be active

## 3.7 Boundary Information Update

In the process of robot fetching spare sensor and relocating them at particular place, topology change takes place in neighborhood of this spare sensor and deployment place. The removal of a redundant sensor may cause its one-hop neighbor's non-redundancy. On the contrary, deploying a sensor at a new place may switch its new neighbor to be redundant. There may be such situation that two robots move to pick up two neighboring spare sensors respectively. However, removing one spare sensor in the topology turns the neighbor spare sensor to an active sensor. In this case, if the active sensor is still removed, a sensing hole would be generated. Before robot takes a sensor, it judges the status of sensor again in order to avoid taking an active sensor in the case referred to above. After a new sensor is deployed, it is selected as representative of the new boundary hole if sensing hole is not fully covered. Boundary nodes relay the "boundary search" message initiated by the new sensor and locally identify the new boundary. Representative of the new sensing

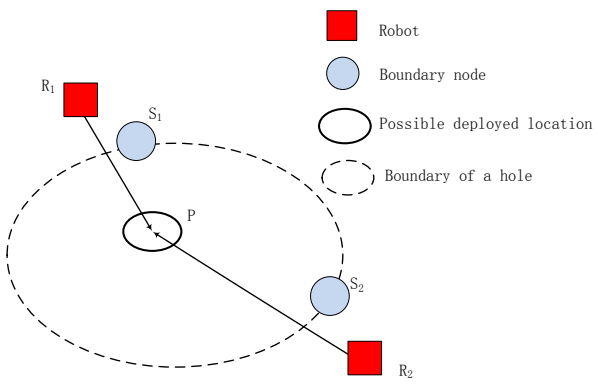



Figure 3.14: Avoidance to duplicate filling one hole

hole obtains possible deployed locations and corresponding progress and shares the information with boundary nodes along the same hole. Boundary nodes then bids to robot once they receive "information collection" message. A new round begins then. The procedure of relocation terminates when there is no more sensing holes or spare sensors, which is the same as the existing algorithm [15].

Fig. 3.13 illustrate an example for the problem mentioned above. $S_1$ and $S_2$ are spare sensors. $S_1$ is in $R_1$'s spare sensor list, while $S_2$ belongs to $R_2$. When $S_2$ is taken by $R_2$, as it is illustrated in Fig. 3.13(b), $S_1$ turns to an active sensor. Since $R_1$ does not know the non-redundancy of $S_1$, it still marches to get $S_1$, as it is shown in Fig. 3.13(b). Before $R_1$ takes $S_1$, it checks $S_1$'s status, which is active in this case. As a consequence, $R_1$ will not take $S_1$ and deletes $R_1$ from its spare sensor list.

Fig. 3.14 shows an example of what we described above. $R_1$ talks to $S_1$ and decide to deploy sensor at $P$, while $R_2$ also intends to fill the hole at $P$ after contacting $S_2$. Since $R_1$ is closer to $P$, it fills the hole first and continues to explore the ROI. When $R_2$ arrives at $P$, it finds there is already one sensor deployed. It would not deploy again and turn to fill other holes. In this way, duplicate deployment can be avoided.

As boundary nodes have information of possible deployed locations along the

whole sensing hole, if two robots contact two boundary nodes respectively, they may pick the same location to deploy a sensor. In this process, it is possible that these two robots do not have a chance to communicate. As a result, two robots will deploy duplicate sensors at the same location. To avoid this kind of duplicate deploying, robot checks if there is already one sensor deployed when it is going to deploy a sensor at certain place.

## 3.8 Comparison of MSR and G-R3S2

The Table. 3.1 shows difference of MSR and G-R3S2 in several aspects:

Table 3.1: Comparison of MSR with G-R3S2

| | G-R3S2 | MSR |
|---|---|---|
| Boundary nodes' knowledge of hole | part of sensing hole | full knowledge of sensing hole |
| Deployed location choosing | Minimize sensing overlap with boundary nodes | Reduce number of hole arcs and minimize sensing overlap with boundary nodes |
| Bid choosing from multiple options | Fill nearest hole | Choose the hole with best $CoP$ |
| Robot cooperation | Not given | Robots within communication range share tasks and spare sensors and compete if same hole or spare sensor is chosen |

# Chapter 4

# Performance Evaluation

In this chapter, we compare G-R3S2 and MSR through extensive simulation. We use the following performance metrics:

1. Coverage Ratio (CR): The ratio of area covered by at least one sensor over the whole area of ROI

2. Total Relocation Time (RT): The time for robots to finish relocation.

3. Moving Distance of Robot (MD): Robot's total travelling distance during the simulation.

4. Number of Robot Moves (RM): The average number of movements made by robots. In the relocation procedure, moving robot may stop at particular places picking up, dropping off sensors or collecting information at grid point. One move of robot is from one stop to its following stop.

5. Message Cost (MC): The average number of messages a sensor transmitted for each robot. Let $nmsg$ denote the total number of messages sensors transmit. $m$ and $n$ represent number of robots and sensors respectively. $MC = nmsg/mn$.

Coverage ratio CR is adopted to illustrate dispersive uniformity. Achieving high CR with same number of sensors demonstrates high dispersive uniformity. Delay times for individual robots are not same, therefore the average delay per robot does not accurately measure the experienced delay for the whole team. Thus we chose to measure here the total time spend by all robots instead. RM measures the number of robot's iterations required to finish the relocation. As additional energy is required for robot's launching, robots' energy consumption is measured by both RM and MD. Sensor's energy consumption is mainly composed of sensor's sensing messages. So MC is adopted to measure sensor's energy expenditure.

Recall the terminating rule of MSR and G-R3S2. All these two algorithms terminate when there is no sensing hole or no spare sensor remained in ROI. In our simulation, we consider a coverage of 98.5% as a good CR, so we these two algorithms terminates when CR reaches 98.5%. As $n$ varies, the two algorithms may reach different terminating rules. For example, when $n$ is set to 400 or 380, it is likely for the robots to achieve a coverage of 98.5% even though not all spare sensors are relocated. Fewer sensors may not be enough to achieve high CR to terminate the relocation process. Then the procedure ends when all redundant sensors are relocated. We slightly modify the measurements here. When we evaluate the impact of $n$ on RT, MD, RM and MC, as different terminating rules may be met, we measure the ratio of each parameter to additional coverage ratio. This kind of measurement judges the average cost with achieving same extra coverage.

## 4.1 Simulation Setup

We implement G-R3S2 and MSR with JBotSim simulator. Experiments were conducted on HP Pavilion PC with Inter Core i7 CPU @ 2.80 GHz and equipped with 6 GB RAM running Windows 7 Enterprise under 64-bit architecture. JBotsim relied

upon JDK 1.7 framework.

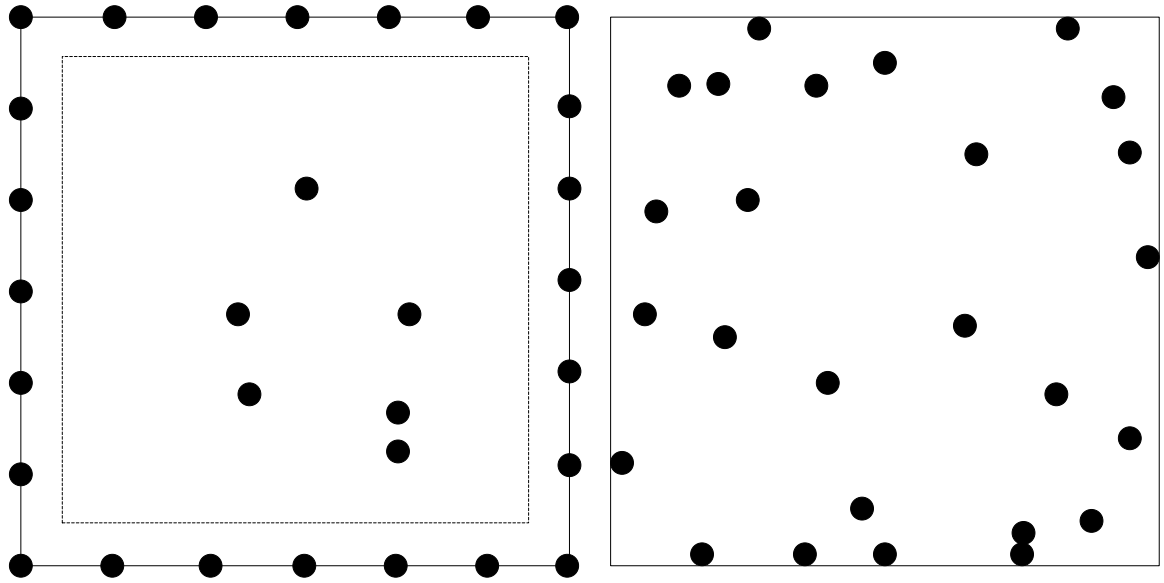
For a fair comparison, we slightly modified the assumption of G-R3S2. We set load of robot in G-R3S2 at 1, the same as MSR. Sensors are initially scattered randomly in ROI. Robots start at specific locations. We set $r_c = 2r_s = 50cm, R_c = 100cm$. In G-R3S2, we choose to use virtual square grid edge of length is 30cm.

Spare sensors are identified by activity scheduling protocol [12]. Sensing holes are locally identified according to the approach in Chapter 3. The size of ROI is $600cm \times 600cm$. Robots moves in the environment in a constant speed 10cm per time unit. Propagation delay of message is 1 time unit.

## 4.2 Experiments

We define two types of initial topology here. The height and width of ROI is denoted as $Height$ and $Width$. For topology-A, as the border of ROI is predetermined, it's easy to pre-deploy $N$ connected sensors along the border. Then scatter $n-N$ sensors in ROI whose coordinates are restricted as $x \in [r_s, Width - r_s], y \in [r_s, Height - r_s]$. For topology-B, all $n$ sensors are randomly scattered in ROI, the coordinates of which $(x, y)$ have restrictions $x \in [0, Width], y \in [0, Height]$. For Topology-A, $N$ sensors are pre-deployed along the border. The distance between two neighboring sensors on the boundary of ROI is shorter than sensors' communication range, so that these sensors on border are all connected. This guarantees all sensing holes can be enclosed by a set of connected sensors. The remaining $n - N$ sensors in randomly scattered in the smaller dashed grid. As it is shown in Fig. 4.1, $n$ sensors are randomly scattered within ROI in Topology-B. We will discuss the performance of MSR and G-R3S2 based on both of these two initial topologies.

To simplify analysis, in simulation, we don't consider node failure during the relocation process, although this kind of problem can be easily solved by our approach.



(a) Topology-A (b) Topology-B

Figure 4.1: Two types of initial topology

We first varied the number of robot $m$, from 1 to 5 to see the impact of $m$ on coverage. $n$ is set at 360. Then we analyze impact of $n$ on coverage ratio by vary $n$ from 300 to 400, with increments of 20, fixing $m$ at 4.

After that, we evaluate the relocation time of the two algorithms. First of all, we study the impact of $m$ on relocation time with setting $m$ from 1 to 5. $n$ is set at 400. Then we analyze the impact of $n$. $n$ varies from 300 to 400 with increments of 20, and $m$ is set at 4. Similarly, we evaluate the impact of $m$ and $n$ on robot total moving distance, robot average number of moves, and message cost. Furthermore, RT, MD, RM and MC to required to achieve certain coverage ratio are measured. Here $m$ is set to 4, and $n$ is set 400. Coverage ratio varies from 90% to 98% with increment of 1%. For each experiment, we run the simulation for 20 times.

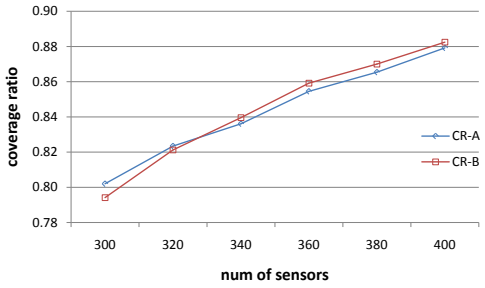



Figure 4.2: Impact of $n$ on initial coverage ratio

## 4.3 Analysis

The following is a detailed analysis of results generated from the simulation of G-R3S2 and MSR. MSR outperforms G-R3S2 in all metrics measured. To be specific, MSR needs less time to finish relocation with less robot energy cost.

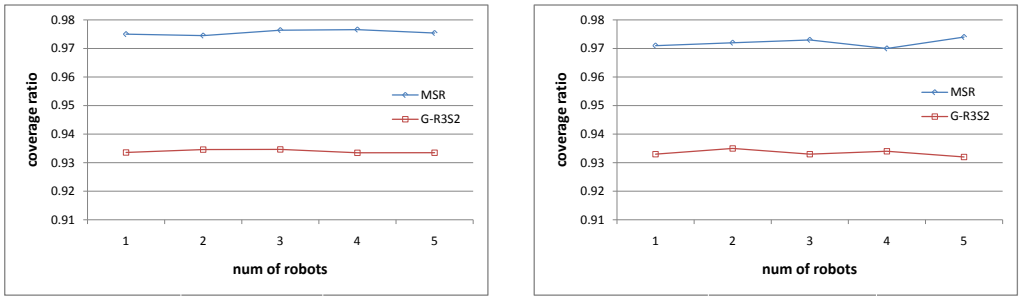
### 4.3.1 Coverage Ratio

#### 4.3.1.1 Initial Coverage Ratio

We first study the impact of $n$ on initial CR. We present two types of initial topologies. As it is illustrated in Fig. 4.2, initial CR increases from 80% up to 88% for both topology-A and topology-B as $n$ varies from 300 to 400. The additional CR created by 20 more sensors is slightly decreasing, since more covering overlap is generated as the density of nodes becomes high. We can see that the difference of initial topology does not have a significant effect on initial coverage ratio.

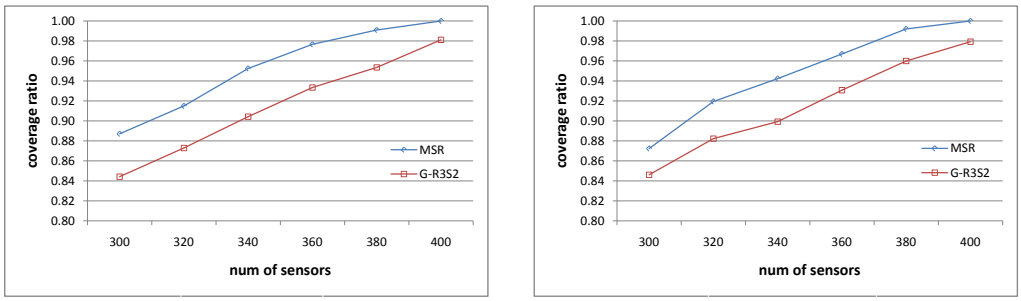
#### 4.3.1.2 Impact of $m$ on Coverage Ratio

We then examine the performance of MSR and G-R3S2 on coverage ratio (CR). The result for how $m$ affects CR is shown in Fig. 4.3. From Fig. 4.3, we can figure out that $m$ does not have a significant influence on CR. With $m$ set to 360, MSR can




(a) Topology-A (b) Topology-B

Figure 4.3: Impact of $m$ on coverage ratio




(a) Topology-A (b) Topology-B

Figure 4.4: Impact of $n$ on coverage ratio

achieve a coverage of 97.5%, while G-R3S2 can only obtain a coverage around 93.5% with same number of sensors. From this figure, we can see that for each kind of initial topology, the coverage ratio of MSR is higher than that of G-R3S2. More robots do not provide higher coverage. Therefore, we can conclude that it is the deployment location choosing strategy that influences the coverage ratio, rather than number of robots.

#### 4.3.1.3 Impact of $n$ on Coverage Ratio

Besides, Fig. 4.4 shows how $n$ affects CR. $m$ is set to 4 and $n$ increases from 300 to 400 with increment of 20. Coverage ratio for MSR and G-R3S2 increase as more nodes are scattered in ROI. $n$ starts at 300, CR of G-R3S2, MSR is around 84% and 88% respectively. As $n$ increases the gap of CR between G-R3S2 and MSR is approximately 5%. For both Topology-A and Topology-B, the increasing trend of CR is similar, so the coverage ratio of these two algorithms is not affected by initial topology. As $n$ increases, the additional coverage ratio acquired by another 20 sensors is decreasing. More sensors scattered in ROI is more likely to generate larger sensing overlap. When $n = 400$, MSR can achieve full coverage. At the same time, CR for G-R3S2 is more than 98% with 400 sensors. Fig. 4.4 shows that for both of Topology-A and Topology-B, CR of MSR is higher than that of G-R3S2 with same number of sensors scattered in ROI.

When relocation process ends, the distribution of MSR is better than G-R3S2 measured by CR. This is caused by difference of deploy policy. G-R3S2 doesn't take size of sensing hole into consideration. As a consequence, robot carrying a sensor fills a hole encountered even though it is small. Here, the robot only fill the nearest hole rather than the largest, which may cause the spare sensor is used to cover a small hole. In MSR, with several candidates holes, robot always seeks a place that can have large additional area covered to deploy sensor, so that the spare sensor can be maximally utilized. Filling larger holes in each iteration, new sensors deployed by robot in MSR are likely to have less sensing overlap with other sensors. Considering the difference of deployment strategy, MSR's sensor deployed strategy is more efficient.

### 4.3.2 Relocation Time

#### 4.3.2.1 Impact of $m$ on Relocation Time

Here we set $n = 400$ and varies $m$ from 1 to 5. As it is illustrated in Fig. 4.5, relocation time required has a decreasing trend as $m$ increases. With a certain work load, more robots can reduce the relocation time. In both of the two algorithms, robots move randomly. More robots increase the probability for sensing hole and spare sensor in ROI to be visited by robot, so spare sensors is likely to be relocated to fill sensing holes in shorter time. From Fig. 4.5, we can see that when $m$ is increased from 1 to 2, for both of the two algorithms, RT is reduced by nearly 60%. This is caused by the moving strategy of these two algorithms, least recently visited (LRV). According to LRV, after the robot leaves one grid point, it is more likely to explore in other areas in ROI. So it may take the robot for a long time until it has a chance to return to this grid point. For example, if the some spare sensors are located right in a small area, after picking up one spare sensor, it would be long time for the robot to come back again to pick up others. Same things happen to sensing holes. If a hole is large that cannot be covered by single spare sensor, it takes a long time for robot to return to this point to continuing filling this hole with a spare sensor. This kind of problem can be properly solved by multiple robots. As each robot has a unique LRV map instructing the next grid point robot heading to, one robot's visiting at a point does not influence other robots' LRV map. As a consequence, it is quite possible that after one robot leaves, another one arrives at the grid point in a short time to do the remaining task, either picking up spare sensors or filling sensing holes. RT required for MSR is less than that for G-R3S2 all the way. This is caused by the different deployed strategy. The efficient deployed strategy of MSR enables robot to gain larger progress for each iteration. Consequently, fewer iterations reduce the whole relocation time.

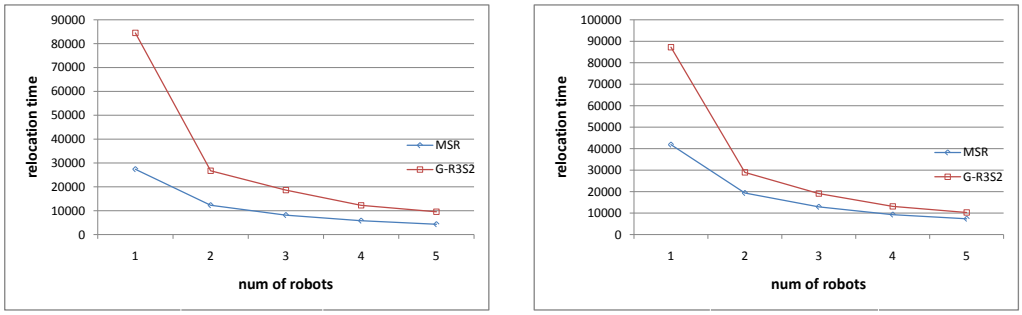


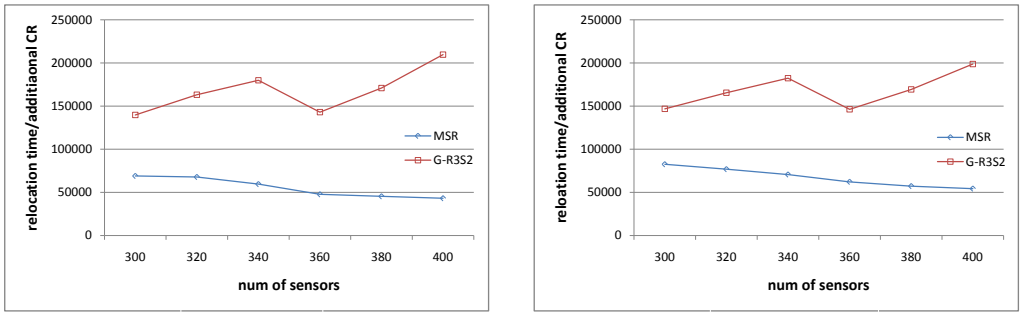

Figure 4.5: Impact of $m$ on relocation time

#### 4.3.2.2 Impact of $n$ on Relocation Time

For $n$ from 300 to 400, RT for MSR is less than that for G-R3S2 all the way according to Fig. 4.6. As $n$ increases, RT for MSR slightly decreases. There are more spare sensors with more sensors initially scattered. When the relocation process terminates, more additional CR can be achieved with more spare sensors available. As $n$ grows, the RT trend of G-R3S2 fluctuates. It rises when $n$ increases from 300 to 340. With more spare sensors to relocate, robot's work load grows. Since robot only achieves limited coverage improvement, the $RT/\Delta CR$ rises. When $n$ reaches 360, the CR improvement significantly increases, so $RT/\Delta CR$ drops at $n = 360$. After that, achieving nearly same CR improvement, robot takes longer time to relocate more sensors. Consequently, $RT/\Delta CR$ increases with 380 and 400 sensors. The trend of $RT/\Delta CR$ is nearly the same for both Topology-A and Topology-B.

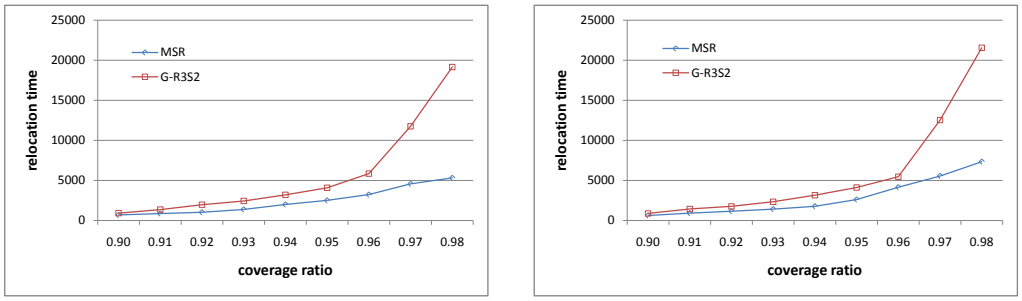
#### 4.3.2.3 RT Required to Achieve Certain Coverage

Then we study the relocation time required to obtain certain CR. Fig. 4.7 presents RT required to gain certain CR. More time is needed for achieving higher CR. To achieve 90% coverage, MSR and G-R3S2 need nearly same amount of time. After that, as CR increases, RT in G-R3S2 is larger than and MSR all the way. With increasing CR, it




(a) Topology-A (b) Topology-B

Figure 4.6: Impact of $n$ on relocation time




(a) Topology-A (b) Topology-B

Figure 4.7: RT required to achieve CR from 90% $\sim$ 98% with $n = 400$

requires more time to achieve additional coverage, especially for G-R3S2. In G-R3S2, taking topology-A as an instance, required relocation time for obtaining additional 1% of coverage varies from 150 all the way up to around 7000. As CR increases, the extra relocation time for additional 1% CR is less for MSR. The reason for this is that robot in MSR has more knowledge about its neighborhood. According to Cop, it may travels a bit longer and fill a much larger hole. Consequently, having sensors to cover more area for each iteration, fewer iterations are required to finish the work. So MSR needs less time to relocate spare sensors to fill sensing holes.

### 4.3.3 Robot Energy Cost

#### 4.3.3.1 Impact of $m$ on Robots' Total Moving Distance

Then we comes to impact of $m$ on robot moving distance. As it is illustrated in Fig. 4.8, MSR's MD has a monotonously decreasing trend. MD indicates total moving distance of robots. Although more robots would travel longer distance in total when they explore the ROI, robot cooperation may improve the performance of one iteration of robot, which reduces robots' iteration times. This counteracts additional MD caused by more robots. For G-R3S2, MD fluctuates as $m$ increases. It reduces significantly from 1 robot to 2 robots. The 30% of MD reduction is caused by robot's moving policy. According to least recently visited (LRV) policy, when robot leaves a grid point, it tends to go to other grids to explore the ROI. So it may take a long time for robot to return this grid point. At the time of robot's leaving a grid point, the work here may not be done (e.g., there are some remaining spare sensors or sensing holes). Consequently, single robot needs to travel long distance to have a chance to return to this grid point. With more robots working in ROI, this problem can be properly solved. As each robot keeps unique LRV map, the work not finished by a robot at a grid can be done by another robot within short time. G-R3S2's MD fluctuates from $m = 2$ to $m = 5$. MSR's MD is always shorter than G-R3S2. MSR's efficient deployed location strategy and robot cooperation policy leads to the less moving distance of robot.

#### 4.3.3.2 Impact of $n$ on Robots' Total Moving Distance

As it is shown in Fig. 4.9, MSR's $MD/\Delta CR$ is much less than G-R3S2's. For instance, with 300 sensors, MSR's $MD/\Delta CR$ is about one half of G-R3S2's. The gap rises as $m$ increases. With more spare sensors available in ROI, robot cooperation of MSR improves the probability for robot to discover spare sensors. Consequently, robots

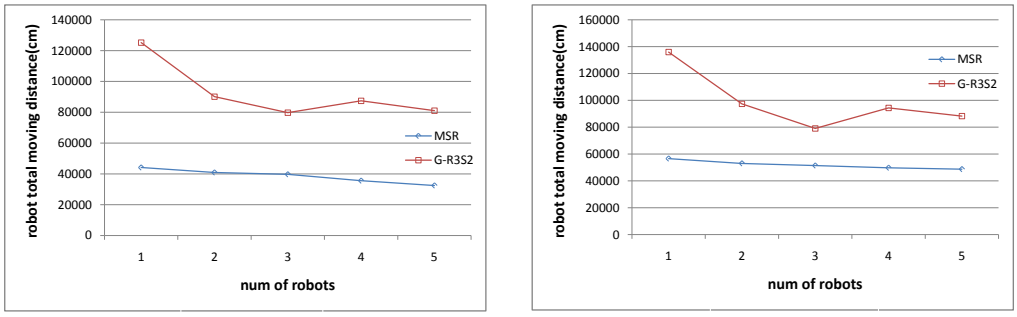



Figure 4.8: Impact of $m$ on robots' total distance

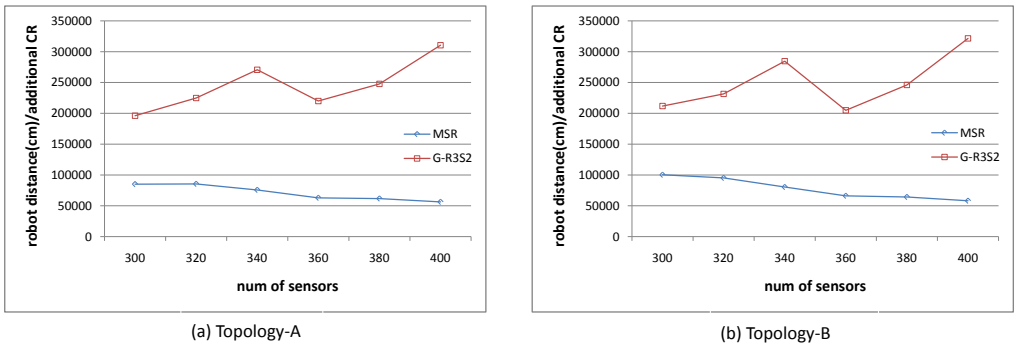
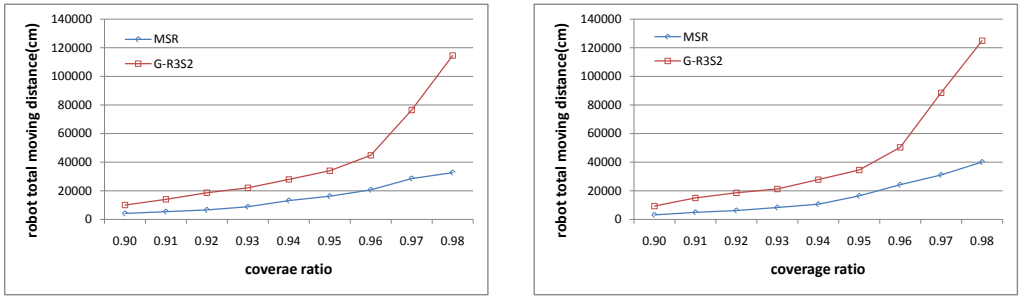



Figure 4.9: Impact of $n$ on robots' total distance

can pick up all spare sensors with travelling shorter distance. When $m$ reaches 380, MSR can achieve a 98.5% while G-R3S2 cannot. The MSR's $MD/\Delta CR$ decreases since more sensors create smaller holes which require fewer spare sensors to fill. So MD is reduced. However, G-R3S2 cannot achieve 98.5% coverage with 380 sensors, so $MD/\Delta CR$ increases compared to $m = 360$.

#### 4.3.3.3 MD Required to Achieve Certain Coverage

In addition, we also examine robot's energy consumption. Fig. 4.10 shows the relation of MD with CR. As it is the same with RT, this figure also presents monotonically increasing trend, with increasing slope. MD in MSR is much shorter than that in




(a) Topology-A

(b) Topology-B

Figure 4.10: MD required to achieve CR from 90% $\sim$ 98% with $n = 400$

G-R3S2, and the gap becomes larger as CR increases which is caused by different deployment strategy. As we referred to above, MSR's deployed location choosing strategy is better than G-R3S2, which reduces robot's iterations. Since robots in G-R3S2 needs to iterate more times to achieve high coverage, it requires more spare sensors for them to fill smaller sensing holes. As CR is high (e.g., 96%), robots need to travel longer distance to look for spare sensors or sensing holes. So robot in G-R3S2 travels much longer distance to achieve 1% extra coverage.

#### 4.3.3.4 Impact of $m$ on Number of Robot Moves

Finally, we study the impact of $m$ on number of robot moves. Robot average number of move represents the number of robots' iterations. As it is the same with RT and MD, Fig. 4.11 also shows a decreasing trend. As the total work load stays the same, each robot has less work to do. For Topology-A, MSR's RM is only one fourth of that in G-R3S2. Due to the limitation of one robot scenario we talked about above, robot in G-R3S2 needs a lot of iterations to finish the work. Meanwhile, when robot in MSR communicates with a boundary node, it has the information of all the boundary. So it can take the best pair of spare sensor and sensing hole according to *CoP*. This helps robot to fill sensing holes in some iterations, which reduce the total time of robot's

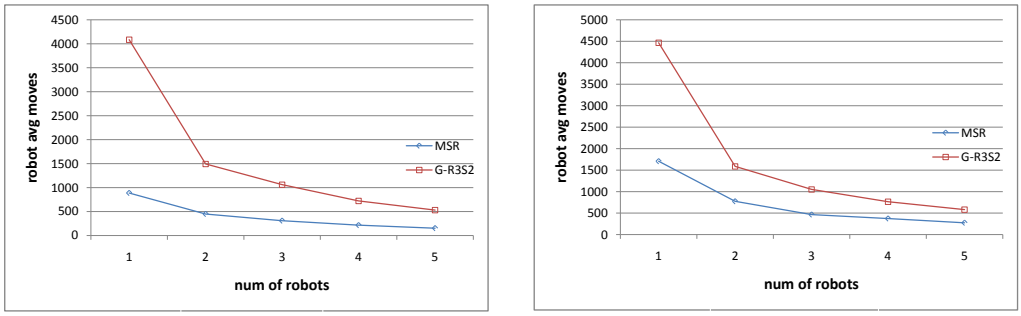



Figure 4.11: Impact of $m$ on robot average moves

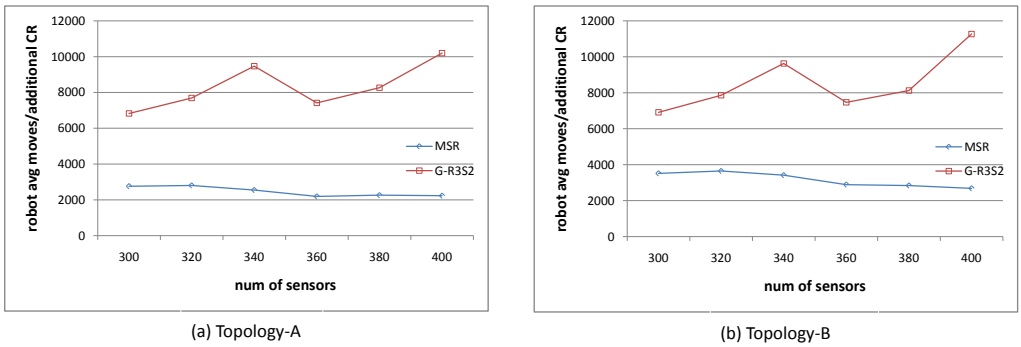
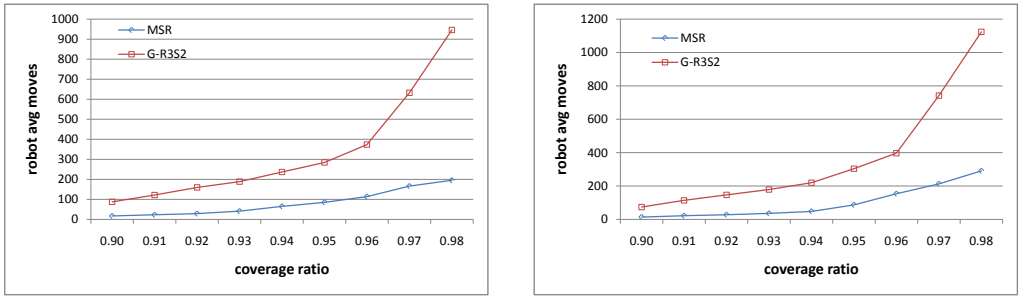



Figure 4.12: Impact of $n$ on robot average moves

iterations.

#### 4.3.3.5 Impact of $n$ on Number of Robot Moves

The trend of $RM/\Delta CR$ shown in Fig. 4.12 is similar as the former two. For G-R3S2, this measurement has large values when $n = 340$ compared to $n = 320$ and $n = 360$. For the situations of 340 sensors, the coverage improvement is low and number of spare sensors is large. As a consequence, it costs more robot iterations to collect all spare sensors. For MSR, robot achieves more additional coverage with more sensors. So the measurement presents a decreasing trend.




(a) Topology-A

(b) Topology-B

Figure 4.13: RM required to achieve CR from 90% $\sim$ 98% with $n = 400$

#### 4.3.3.6 RM Required to Achieve Certain Coverage

Fig. 4.13 illustrates the relation of RM and CR. We notice that RM in MSR is so small that only around 60 moves is required to achieve 98% coverage in Fig. 4.13. Obtaining same CR, MSR-R needs 200 moves and G-R3S2 needs 950. This is caused by moving pattern of G-R3S2 and MSR, which stops robots at grid point to let them judge whether there is spare sensor to take or sensing hole to fill. Also, robot at grid points determines which direction to go according to LRV we referred to in 2.

As robot's energy consumption is measured by RM and MD, robot energy for MSR is much less than G-R3S2.

### 4.3.4 Message Cost

Since sensor's main energy consumption is transmitting messages, MC can be used to measure sensor's energy cost. The cases sensors transmitting messages are described as follows:

For G-R3S2, when robot arrives at a grid point, it collects information in neighborhood by broadcasting a "information collection" message. Receiving this kind of message, boundary nodes replies with "best deployed location" in its own vision.

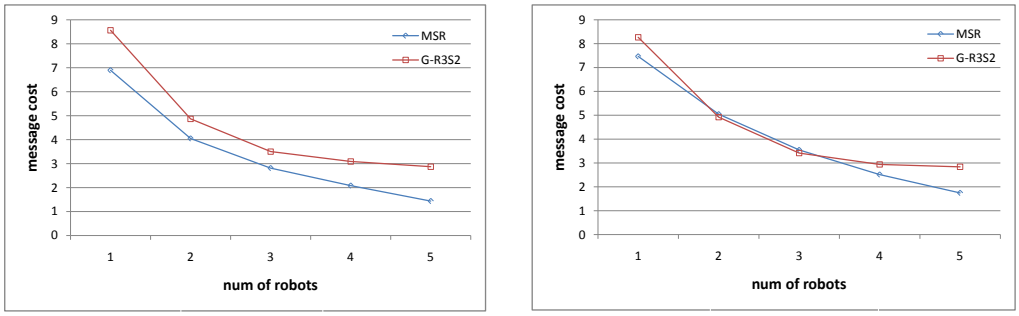



Figure 4.14: Impact of $m$ on robot average moves

Furthermore, spare sensors also report their information to connected robot. For MSR, in addition to the message cost as referred to above in G-R3S2, sensors also locally identify sensing holes by transmitting messages. The message cost for "Hello" message is omitted since the message cost for this is nearly the same for these two algorithms.

#### 4.3.4.1 Impact of $m$ on Message Cost

As it is illustrated in Fig. 4.14, in Topology-A, MSR's message cost is lower than G-R3S2's. Meanwhile, MC is nearly the same for these two algorithms when $m = 2, 3$. MC for G-R3S2 is higher with other values of $m$. Although sensors in MSR locally identify sensing holes by sending messages, the message cost is still lower than G-R3S2 in most cases. As we referred above, the search work for robot in G-R3S2 is much more than MSR. In searching procedure, boundary sensors report possible locations to neighboring robot at grid point. The long searching process increases the message cost of G-R3S2. More robots can reduce the searching procedure so that the message cost is saved.

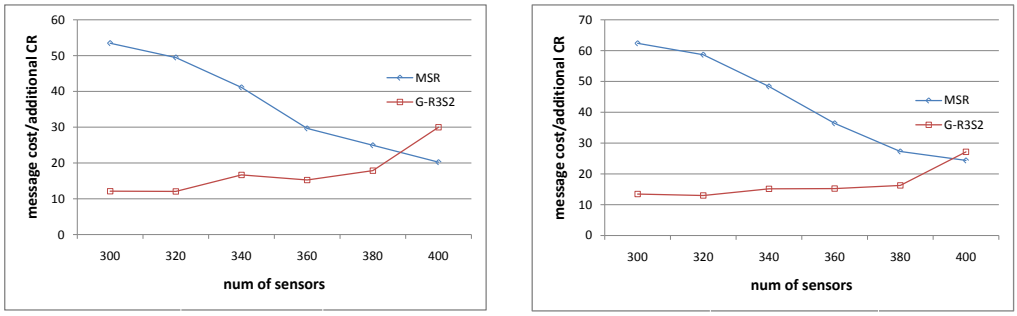


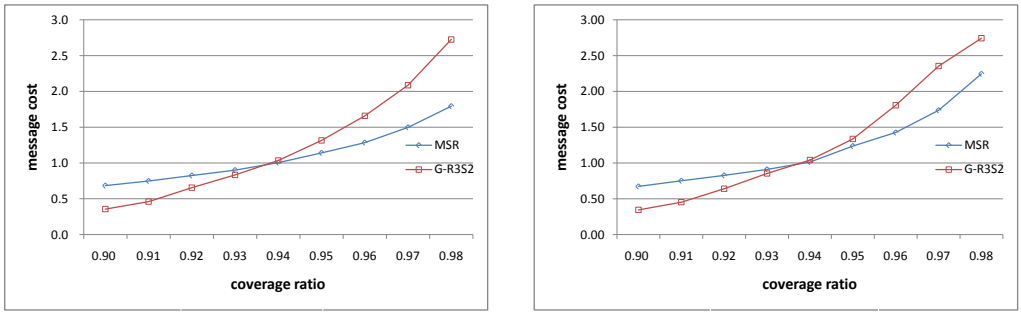

Figure 4.15: Impact of $n$ on robot average moves

#### 4.3.4.2 Impact of $n$ on Message Cost

As it is shown in Fig. 4.15, $MC/\Delta CR$ for MSR increases with accumulating number of sensors. More sensors indicates more spare sensors. For each robot's iteration of relocation, when redundant sensor is deployed, the new boundary hole would be identified. In the boundary updating process, messages traverse the new hole. Consequently, the more sensors to be relocated, the more message cost generated. For G-R3S2, since there is no message cost for boundary update, the measured parameter presents a decreasing trend with accumulating number of sensors. This is caused by the increasing additional CR gained.

#### 4.3.4.3 MC Required to Achieve Certain Coverage

Fig. 4.16 illustrates required message cost to achieve certain coverage. The figure presents a monotonously increasing trend. MC required G-R3S2 to achieve low coverage (e.g., from 90% to 93%) is less than MSR, which is caused by MSR's the extra message cost in identifying sensing holes. However, for high coverage (e.g., 95% or more), the message cost of G-R3S2 exceeds MSR. Due to the inefficiency of deployed strategy and limited vision of robot, robot in G-R3S2 searches long way to find a task. During this process, sensors transmits a lot of messages to neighboring robot.




(a) Topology-A (b) Topology-B

Figure 4.16: MC required to achieve CR from $90\% \sim 98\%$ with $n = 400$

As coverage ratio increases, the amount of MC to gain additional 1% coverage rises.

# Chapter 5

# Conclusions and Future Work

We proposed a localized carrier-based sensor relocation algorithm MSR, with single-robot and multi-robot scenarios. First of all, boundary sensors cooperate to identify the size and shape of a sensing hole and choose one boundary sensor as a representative. The representative is responsible for collecting information about each hole. With information of all boundary sensors on the hole, representative calculates several possible deployed locations and shares the locations with boundary sensors by multi-hop messages. This representative is designed to save message cost in hole identification process. Then we proposed to deploy new sensor at a place that enables it to cover three consecutive intersections. Compared to the deployed location strategy in R3S2, this strategy is less likely to generate small holes, which fills sensing hole with fewer sensors. In addition, if more than one spare sensor or sensing hole are available, we made robot choose the task with smallest $CoP$, while in R3S2, robot chooses to fill the nearest hole. The consideration of progress for filling sensing holes in MSR makes robot's relocation more efficient. We adopted the same robot moving strategy, least recently visited (LRV) as the R3S2. Robot cooperation was designed to enlarge robots' vision and present each robot with more choices of spare sensors and sensing holes so that robots may choose more suitable spare sensors and sensing holes. This

provides possibility to improve relocation efficiency of each robot's iteration.

Compared to the only existing localized algorithm R3S2, the main novelties are coordination among robots (using a method similar to [23]), and an improved scheme for finding a good location inside the hole to place a new sensor. Furthermore, with a list of tasks available, robot's choice based on $CoP$ provided better performance. Our simulation results indicate MSR outperforms R3S2 in various metrics. It reduces relocation delay, robot energy costs and sensor energy costs.

Future work could consider following aspects: As robot's random moving strategy causes long exploring process, MSR can be improved by adopting other robot moving strategies. It could also benefit from clustering sensors and/or robots [1] and reducing the communication cost by emphasizing communications from cluster heads. Further, approaches to overcome problem caused by unbounded sensing hole, other than modifying initial topology, are also needed. For instance, sensors could be first deployed optimally [20] and then relocation applied reacting to failures. In MSR, robots are relocating only fully redundant spare sensors. It can be modified to allow moving sensors who are mostly but not fully redundant, from an area with little contribution in monitoring area, to an area with much higher coverage contribution. A probabilistic method that applies such relocation is described in [30]. A new method may incorporate probabilistic picking up and dropping sensors with the ideas presented in MSR. We assumed a simplified model of circular area coverage. Probabilistic area coverage can be considered, using a model such as in [5].

# Bibliography


[1] Zaher Al Aghbari, Ibrahim Kamel, and Walid Elbaroni. Energy-efficient distributed wireless sensor network scheme for cluster detection. *International Journal of Parallel, Emergent and Distributed Systems*, 28(1):1–28, 2013.

[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] Chih-Yung Chang, Chao-Tsun Chang, Yu-Chieh Chen, and Hsu-Ruey Chang. Obstacle-resistant deployment algorithms for wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 58(6):2925–2941, 2009.

[4] Chih-Yung Chang, Jang-Ping Sheu, Yu-Chieh Chen, and Sheng-Wen Chang. An obstacle-free and power-efficient deployment algorithm for wireless sensor networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 39(4):795–806, 2009.

[5] Pingsheng Chen and Weidong Hu. Sleep-wake up scheduling with probabilistic coverage model in sensor networks. *International Journal of Parallel, Emergent and Distributed Systems*, 29(1):1–12, 2014.

[6] Predrag M Ćirković and Zoran Lukić. Self-deploying wireless sensor network. *17th Telecommunication forum*, pages 1490–1493, 2009.

[7] Brian Coltin and Manuela Veloso. Mobile robot task allocation in hybrid wireless sensor networks. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2932–2937. IEEE, 2010.

[8] Rafael Falcon, Xu Li, and Amiya Nayak. Carrier-based coverage augmentation in wireless sensor and robot networks. In *IEEE 30th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 234–239. IEEE, 2010.

[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 *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2010.

[10] Greg Fletcher, Xu Li, Amiya Nayak, and Ivan Stojmenovic. Back-tracking based sensor deployment by a robot team. In *7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–9. IEEE, 2010.

[11] Greg Fletcher, Xu Li, Amiya Nayak, and Ivan Stojmenovic. Randomized robot-assisted relocation of sensors for coverage repair in wireless sensor networks. In *IEEE 72nd Vehicular Technology Conference Fall*, pages 1–5. IEEE, 2010.

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

[13] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[14] Ahmed M Khedr and Hager Ramadan. Effective sensor relocation technique in mobile sensor networks. *International Journal of Computer Networks & Communications (IJCNC)*, 3(1):204–217, 2011.

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

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

[17] Xu Li, Hannes Frey, Nicola Santoro, and Ivan Stojmenovic. Strictly localized sensor self-deployment for optimal focused coverage. *IEEE Transactions on Mobile Computing*, 10(11):1520–1533, 2011.

[18] 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.

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

[20] Zhuofan Liao, Jianxin Wang, Shigeng Zhang, and Xi Zhang. A deterministic sensor placement scheme for full coverage and connectivity without boundary effect in wireless sensor networks. *Ad Hoc & Sensor Wireless Networks*, 19(3-4):327–351, 2013.

[21] SH Liu, Yu Zhang, HY Wu, and Jie Liu. Multi-robot task allocation based on swarm intelligence. *Journal of Jilin University*, 40(1):123–129, 2010.

[22] 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.

[23] Ivan Mezei, Veljko Malbasa, and Ivan Stojmenovic. Greedy extension of localized auction based protocols for wireless actuator task assignment. *Ad Hoc & Sensor Wireless Networks*, 17(1-2):73–85, 2012.

[24] Yipeng Qu and Stavros V Georgakopoulos. A centralized algorithm for prolonging the lifetime of wireless sensor networks using particle swarm optimization. In *IEEE 13th Annual Wireless and Microwave Technology Conference (WAMICON)*, pages 1–6. IEEE, 2012.

[25] Yipeng Qu and SV Georgakopoulos. Relocation of wireless sensor network nodes using a genetic algorithm. In *IEEE 12th Annual Wireless and Microwave Technology Conference (WAMICON)*, pages 1–5. IEEE, 2011.

[26] Claudio Rossi, Leyre Aldama, and Antonio Barrientos. Simultaneous task subdivision and allocation for teams of heterogeneous robots. In *IEEE International Conference on Robotics and Automation*, pages 946–951. IEEE, 2009.

[27] Ridha Soua, Leila Saidane, and Pascale Minet. Sensors deployment enhancement by a mobile robot in wireless sensor networks. In *Ninth International Conference on Networks (ICN)*, pages 121–126. IEEE, 2010.

[28] Guiling Wang, Guohong Cao, Piotr Berman, and Thomas F La Porta. Bidding protocols for deploying mobile sensors. *IEEE Transactions on Mobile Computing*, 6(5):563–576, 2007.

[29] Guiling Wang, Guohong Cao, Tom La Porta, and Wensheng Zhang. Sensor relocation in mobile sensor networks. In *INFOCOM. 24th Annual Joint Confer-*

*ence of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, pages 2302–2312. IEEE, 2005.

[30] Yuan Wang, Ahmed Barnawi, Rodrigo Francisco de Mello, and Ivan Stojmenovic. Localized ant colony of robots for redeployment in wireless sensor networks. *Journal of Multiple-Valued Logic and Soft Computing*, to appear.

[31] Wei Zha and Wee Keong Ng. Prolonging lifespan of sensor networks using redundant nodes. *Ad Hoc & sensor wireless networks*, 19(3-4):1–19, 2013.