國 立 交 通 大 學

電信工程學系

碩 士 論 文

無線感測器自動化網路佈署協定設計與實作
Coverage-aware Sensor Deployment Schemes and Implementation of an Automated Home Monitoring Network

研究生：劉葳庭

指導教授：林亭佑 博士

中 華 民 國 99 年 1 月

無線感測器自動化網路佈署協定設計與實作
Coverage-aware Sensor Deployment Schemes and Implementation of an Automated Home Monitoring Network

研　究　生：劉葳庭　　　Student：Wei-Ting Liu

指導教授：林亭佑　　　Advisor：Ting-Yu Lin

國　立　交　通　大　學

電信工程學系

碩　士　論　文

A Thesis

Submitted to Department of Communication Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Communication Engineering

June 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年一月

無線感測器自動化網路佈署協定設計與實作

學生：劉葳庭　　　　　　　　　　　　　指導教授：林亭佑 博士

國立交通大學電信工程學系（研究所）碩士班

摘　　要

近年來無線感測網路的技術蓬勃發展，其應用也推陳出新，事實上，對於無線感測網路而言，如何提供有效的感測覆蓋率是決定網路運作效率的重要因素。在這份三年的計畫中，我們致力於設計居家智慧型無線感測網路，我們在感測器上配置行動裝置使其具有行動能力，並針對居家環境設計一自動化感知傳測器佈署協定(Coverage-Aware Sensor Automation protocol，以下簡稱 CASA)，實現此居家型高智慧網路，藉由動態感測器的自動佈署，以提供使用者所需的感測覆蓋率。此外，有別於其他先前的研究，我們所設計的 CASA 協定允許網路中同時使用齊性或非齊性感測器，也就是說，CASA 協定亦適用於感測範圍(sensing range)不同的感測器，在使用上具有較大的彈性。事實上，CASA 協定主要由 EVFA-B、CFPP、SSOA 這三個機制構成。EVFA-B 會針對我們設計的距離門檻值 d_{th} 使感測器彼此之間運作引力或斥力，其合力結果會將感測器逐漸推向合適的位置，以強化網路中的感測覆蓋率。為了達到高品質的感測覆蓋率，我們研究出 EVFA-B 中所使用的環境參數與網路拓樸有相當大的關係，例如:監控面積大小與網路中的感測器數量，我們期望 EVFA-B 能提供有效的自動化佈署。此外，我們發現當感測器重新佈署時，在移動的過程中可能會有碰撞問題發生，因此我們規劃 CFPP 演算法，針對每一台感測器的移動路徑，事先偵測潛在的碰撞發生地點，並重新調整感測器的移動時程，藉此避免碰撞發生。除此之外，當網路中有某些感測器發生故障或電力不足的情形，我們設計 SSOA 演算法進行局部的修復行動，也就是說，當有感測破洞發生(sensing void)時，SSOA 會選擇此破洞周圍某些合適的感測器去修補它，而不是使用 EVFA-B 重新佈署整個網路，藉此有效節省電力消耗。除此之外，我們發現如何選擇合適的救援感測器，事實上屬於 Maximum-Weight Clique Problem(以下簡稱 MWCP)，此問題被公認為 NP-hard，我們將 MWCP 簡化(reduce)為選擇救援感測器的問題，發現我們的 weight 值可能有正有負，然而目前能解決 MWCP 的演算法只考慮 weight 值恆正的情況。因此，最終我們定義救援感測器選擇的問題時，只考慮 weight 值恆正的情況，如此一來才存在有效率的 polynomial-time 演算法，而 weight 值為負的情況就使用 EVFA-B 代為解決，藉由此合作機制，CASA 協定可以達到有效率的覆蓋率要求。在真實的環境中使用嵌入式系統運行 CASA 協定，藉此設計一個可以容許感測器故障，藉由自動化佈署以延長網路使用壽命的居家智慧型監控網路(MoNet)。此外我們會藉由觀察覆蓋率達成率、網路自我修復能力、移動所耗費的電力，並實地模擬當緊急災害的發生時，MoNet 的事件回報率，藉此估測 CASA 協定的效率。

Coverage-aware Sensor Deployment Schemes and Implementation of an Automated Home Monitoring Network

Student：Wei-Ting Liu　　Advisor：Dr. Ting-Yu Lin

Department（Institute）of Communication Engineering
National Chiao Tung University

ABSTRACT

For the wireless sensor network (WSN) to operate successfully, a critical issue is to provide sufficient sensing coverage. In this thesis, we target on the home environment and deal with both the homogeneous (having identical sensing radius) and heterogeneous sensors (having different sensing ranges) equipped with locomotion facilities to assist in the sensor self-deployment. A coverage-aware sensor automation (CASA) protocol is proposed to realize an automated home monitoring network. Three centralized algorithms are included in the CASA protocol suite: EVFA-B, CFPP, and SSOA. Unlike most previous works that tackle the deployment problem only partially, we intend to address the sensor deployment-related problems in a holistic manner. The enhanced virtual forces algorithm with boundary forces (EVFA-B) exerts weighted attractive and repulsive forces on each sensor based on predefined distance thresholds. The resultant forces then guide the sensors to their suitable positions with the objective of enhancing the sensing coverage (after a possibly random placement of sensors). To achieve high coverage ratio, we prove that good choices for the associated weight constants greatly depend on sensor population and monitored area size, while independent of sensing radius. When sensors move around to self-deploy, the collision-free path planning (CFPP) algorithm, based on geometric formulations, comes into play by carefully scheduling the moving paths to avoid sensors colliding each other. Furthermore, in the presence of sensor power depletions and/or unexpected failures, our sensor self-organizing algorithm (SSOA) is activated to perform local repair by repositioning sensors around the sensing void (uncovered area). This capability of local recovery is advantageous in terms of saving the communication and moving energies. Selection of local rescue sensors with mixed negative and positive weights is NP-hard, and can be reduced from the *maximum-weight clique problem*. We resolve this selection problem by considering only positive weights (leaving the negative weights to be handled by EVFA-B), so that efficient polynomial-time computation can be utilized. In the case that local repairing is unable to provide required sensing coverage, SSOA invokes EVFA-B to globally redeploy sensors. As a result, adequate sensing coverage can be maintained even in the face of sensor node failures, effectively extending network functioning time. Performance of the proposed sensor deployment strategies is evaluated in terms of surveillance coverage, network self-healing competence, and moving energy consumption. We also implement our CASA protocol suite in a real-life home monitoring network (MoNet) to demonstrate the protocol feasibility and validate the MoNet detection capability of emergency events.

誌謝

研究所生涯終於告一段落，這一切都要感謝許多人對我的提攜與鼓勵。首先要感謝我的指導教授－林亭佑博士，在我碩士生涯的過程中，老師悉心的教導並指點我研究的方向，使得我在這段日子中獲益匪淺，尤其是在準備論文與口試期間，多虧有老師的協助與指導，使得本論文能夠更臻完整與嚴謹。

再來，要感謝冠勳學長在硬體實作上給我的協助，感謝學長教導我許多實作相關的知識與技術，也要感謝冠樺學弟、宗欽學弟、修銘學弟與柏齊學弟在各式嵌入式競賽中的付出與努力；另外要感謝Charka，對於此論文的諸多貢獻與付出，也要謝謝實驗室同學們 – 境宜、昆儒、承穎，在生活中給我的幫助與鼓勵；此外，也感謝其他所有 Bun Lab 的成員，有你們的陪伴讓我的碩士生活增添不少色彩。

最後，要感謝我的父母親，從小對我的悉心栽培與諄諄教誨，給予我無限的鼓勵與協助，讓我能無後顧之憂專心致志於研究上，並且達成目標，謹以此文獻給我摯愛的雙親。

誌於 2010.01 新竹 交大

葳庭

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Advances of micro-electromechanical system (MEMS), sensing technology, and wireless communication have significantly encouraged the development of wireless sensor networks (WSNs) in the past decade. A WSN is widely used for habitat and environmental surveillance, medical application (with the purpose of improving quality of health care), agricultural assistance, and as solutions to military problems [5, 11, 15, 16]. Several experimental testbeds are also implemented to investigate various aspects of WSN-related performance issues [10,19,21,23]. Since different environments usually guide WSN studies to distinct research directions and design considerations, it is necessary to firstly define the target environment under investigation. In this thesis, we focus on the indoor home environment, as depicted in Fig. 1.1. To furnish the home with monitoring capability, one possibility could be embedding a secret compartment under the roof, and deploying smart sensors inside the double-deck structure on the ceiling. For a successful home surveillance, providing sufficient sensing coverage is essential. Manual placement of static sensors involves labor effort (reaching the ceiling to perform the planned deployment) and lacks network self-healing competence (when faulty sensors occur). Thanks to the availability of motion facilities, we consider smart sensors with mobility capability to accomplish self-deployment after an initial random placement of sensors. Furthermore, since sensing devices are inherently unreliable, faulty sensors due to power depletions or unexpected

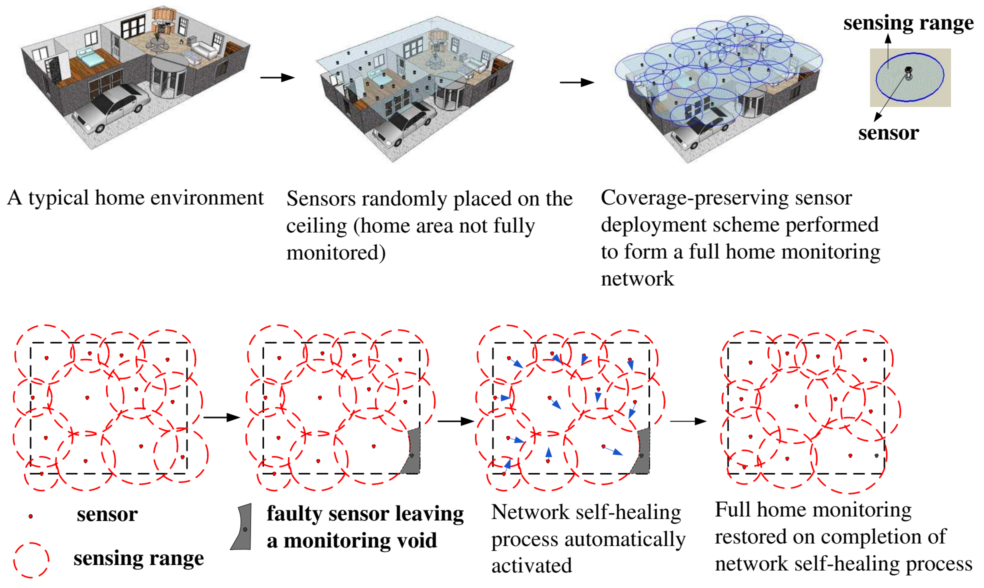

sensing range
sensor
A typical home environment
Sensors randomly placed on the ceiling (home area not fully monitored)
Coverage-preserving sensor deployment scheme performed to form a full home monitoring network
sensor
sensing range
faulty sensor leaving a monitoring void
Network self-healing process automatically activated
Full home monitoring restored on completion of network self-healing process

Figure 1.1: Illustration of an automated home monitoring network, and the importance of (movement-assisted) network self-healing capability to tolerate sensor faults (no need to deploy new sensors).

errors may occur over time, leaving monitoring voids (uncovered sensing holes). With the movement ability, instead of replacing faulty sensors with new ones, those smart sensors reposition themselves to restore the sensing coverage, as illustrated in Fig. 1.1. According to the above descriptions, several deployment-related issues need be addressed. First, a ***coverage-aware sensor deployment scheme*** should be developed to ensure sufficient sensing coverage. Second, when sensors reposition themselves, a ***collision-free route scheduling strategy*** is required to prevent sensors colliding each other. Third, in the face of sensing node failures, a ***sensor self-organizing mechanism*** need be devised to efficiently recover the sensing void and restore the required sensing coverage. In this work, we do ***not*** intend to study the energy-conserving sensor communication behavior (though we try to reduce the moving energy by keeping sensors from moving far away when performing self-deployment), ***nor*** the issue of required amount of sensors to achieve certain degree of sensing coverage. Rather, given any number of sensors, we investigate the deployment-related problems and propose a coverage-aware sensor automation (CASA, which means "home" in Spanish) protocol including the aforementioned three deployment-related designs, with the objective of providing/maintaining high sensing coverage. Our ultimate goal is to realize an automated home monitoring network, so that detection applications of various emergence events can be practically implemented.

The remainder of this thesis is organized as follows. Chapter 2 reviews several prior research efforts and summarizes our unique contributions. In Chapter 3, we introduce the coverage-aware sensor automation (CASA) protocol and provide the environmental assumptions made by the protocol. The proposed CASA protocol consists of three closely-related algorithms to address the sensor deployment scheme (EVFA-B), collision-free route planning (CFPP), and sensor self-organizing mechanism (SSOA), respectively. Chapter 4, Chapter 5, and Chapter 6 elaborate on the detailed operations of EVFA-B, CFPP, and SSOA separately. Chapter 7 presents the performance and comparison results, while Chapter 8 reports our prototype of a home monitoring network (MoNet) and demonstrates the detection capability of CASA-enabled MoNet. Finally, we draw our concluding remarks in Chapter 9.

Chapter 2

Prior work

Depending on the target applications, earlier studies in WSNs generally focus on either outdoor large-scale environments, where planned sensor deployment is difficult, or indoor small-scale monitoring zones, where sensor deployment mechanism is feasible and beneficial. For large-scale WSNs, several works have been proposed to address the energy conservation issue [14, 22, 26, 29, 30]. Given sufficient number of sensors randomly deployed (scattered) over the monitoring field to ensure a certain degree of redundancy in sensing coverage, those proposals design node working schedules such that sensors can rotate between active and sleep modes. The objective of those proposed working schedules (node-scheduling protocols) is to achieve power conservation (prolonging system lifetime), while preserving reasonable sensing coverage and network connectivity.

For the monitoring environments where planned sensor deployment is possible, various static deployment strategies have been introduced to enhance the surveillance coverage [7, 8, 12, 25, 27]. In this kind of research studies, one commonly considered metric is to minimize the number of sensors required to achieve a certain sensing coverage. Due to different sensor capabilities (e.g., distinct attainable sensing/detection ranges) and manufacturing expenses, this metric is sometimes transformed into minimizing/optimizing the required total device cost for those deployed sensors, making this research subject more interesting yet challenging [7,25]. However, such static deployment involves manual sensor placement/installation, and is incapable of dynamically repairing sensing voids (uncovered areas) in the presence of unexpected sensor failures.

Consequently, a number of research efforts have explored the movement-assisted sensor deployment techniques by utilizing mobile sensors to enhance the sensing coverage after an initial random placement of sensors [24,28,31]. With the motion facilities equipped at the sensing devices, sensors can move around to self-deploy. Given any number of randomly placed sensors, in [31], the authors present a centralized force-guided algorithm, inspired by the disk packing theory and virtual force field concept from robotics, to establish motion paths for sensors. Assuming there exists a powerful clusterhead, capable of communicating with all sensors and obtaining sensor locations, the proposed algorithm evaluates all attractive and repulsive forces and obtains the resultant force exerted on each sensor. The computed resultant force then directs the sensor to move to a desired position. Also utilizing mobile sensors, the authors in [24] introduce a distributed sensor self-deployment scheme. They suggest to firstly identify the coverage holes (sensing voids) based on Voronoi diagram, and then propose three algorithms (choices) to guide sensor movements toward the detected holes. However, accurate Voronoi polygon constructions are not always possible to achieve, due to unevenly distributed sensors with limited communication distances. Therefore some optimization heuristic is needed to prevent sensors from moving too far and keep a reasonable number of total movements, further complicating the deployment computations. Furthermore, since the termination condition for the Voronoi-based deployment strategy is coverage, for a monitoring environment with sensor number much larger than necessary, unbalanced sensor distribution (some areas are much more highly populated than other areas, even with an overall sensing coverage required) is likely to occur. As a result, the authors in [28] develop a scan-based movement-assisted sensor deployment (SMART) method to address the unbalanced problem. Instead of tackling the deployment problem directly, SMART focuses on sensor load balancing by using 2D scanning and dimension exchanges to achieve a balanced network state. As claimed by the authors, SMART can operate on top of existing sensor deployment schemes, and produces good performance especially for unevenly distributed WSNs. The aforementioned movement-assisted sensor deployment techniques all consider homogeneous sensors (with equal sensing/detection radius), and no specific route planning strategies are available to perform collision-free movements between sensors.

We observe that most previous works explore the sensor deployment problem only partially, leaving issues such as heterogeneous sensors (with different sensing ranges), sensor moving path scheduling, and locally recovering sensing holes (caused by sensor failures) unaddressed. However, in practice, those closely-related deployment isues should be resolved as a complete protocol set to achieve an operative WSN with high detection capability. In light of this, we investigate the movement-assisted sensor deployment subject by considering those deployment-related problems in a holistic manner. A coverage-aware sensor automation (CASA) protocol suite is proposed to address the global sensor deployment scheme (EVFA-B), the sensor moving path planning (CFPP), and sensing coverage recovery in the presence of sensor failures (SSOA). We summarize our unique contributions as follows. First, ***we develop the enhanced virtual forces algorithm with boundary forces (EVFA-B) based on the concept of potential field and disk packing theory***. Though sharing similar idea of virtual forces with [31], our EVFA-B deals with both the homogeneous and heterogeneous sensors, while [31] only discusses the case of homogeneous sensors, where a global distance threshold value is adopted in determining whether an attractive (with weight constant w_a) or repulsive (with weight constant w_r) force should be applied on a sensor. However, in realistic settings, where varying sensing distances are common, *the distance threshold (determining the desirable sensing overlapping degree) should be selected on a node-pair basis, instead of being set* *globally*. In addition, since the observed environment is usually in a bounded area, *our EVFA-B incorporates the boundary force (with weight constant w_b) as a kind of repulsive force from the boundaries to keep sensors staying inside the monitoring area*. Since the boundary force is considered as a type of repulsive force, we use the same value for w_r and w_b. In [31], no boundary force is modeled, and no specific design guidelines are available for determining suitable w_a and w_r ($=w_b$) weight constants. The authors only suggest to select $w_r >> w_a$. However, we discover that arbitrary settings (even satisfying $w_r >> w_a$) do not always yield desirable sensing coverage. Motivated by the observations, we investigate and prove that *good choices for w_a and w_r ($=w_b$) greatly depend on sensor population and monitored area dimensions, while independent of sensing radius*. Second, ***we propose a collision-free path planning (CFPP) strategy, based on geometric formulations, to avoid sensors colliding each other when performing self-deployment***. This route scheduling is necessary in order to achieve effective sensor deployment in real environments. Third, ***the sensor self-organizing algorithm (SSOA) is devised to provide network self-healing (automated fault recovery) capability***, which most previous sensor deployment protocols do not handle. Fourth, we observe that most existing works do not have a real-life testbed to demonstrate their proposed protocols/algorithms. In this work, ***we implement a home monitoring network (MoNet)***, based on embedded platforms, sensing components, communication modules, and motion devices, ***to validate the proposed CASA protocol***.

Chapter 3

Coverage-aware Sensor Automation (CASA) Protocol

Three deployment-related mechanisms are incorporated in our CASA protocol set: EVFA-B, CFPP, and SSOA. The detailed operations of respective mechanism, with the objective of enhancing/preserving/recovering the sensing coverage for a home environment, are elaborated in Chapter 4, Chapter 5, and Chapter 6, respectively. Below we summarize the environmental assumptions made in this work.

(A1) There exists a powerful clusterhead responsible for performing centralized computations. All sensors are able to communicate with the clusterhead via single-hop or multi-hop wireless transmissions.

(A2) Sensors have the isotropic sensing shape and the binary sensing/detection behavior, in which an event is detected (not detected) by a sensor with complete certainty if this event occurs inside (outside) its sensing radius. Both the homogeneous (having identical sensing range) and heterogeneous (having varying sensing ranges) sensors are allowed in our model. Information of respective sensing ranges is provided by all sensors and made available at the clusterhead for deployment-related computations.

(A3) We adopt the discrete coordination system, in which the monitoring area (sensing field) is represented by a 2D grid network. Locations of all sensors are obtained via the pre-deployed RFID platform or some existing localization technique, and constantly updated to the clusterhead. Neighboring nodes under the adopted coordination system are defined as sensors within the sensing range (r_s), which is normally much smaller than the radio communication distance (r_c). Without loss of generality, we assume that $r_c > 2r_s$ in our model. According to the derivations in [14, 30], if the radio communication range (r_c) is at least twice the sensing radius (r_s), complete coverage of a convex area implies connectivity among the working set of sensor nodes. Consequently, in this work, we only deal with the sensing coverage, and network connectivity follows accordingly.

Chapter 4

Enhanced Virtual Forces Algorithm with Boundary Forces (EVFA-B)

The concept of virtual forces is inspired by the combined idea of potential field and disk packing theory [9,13]. Each sensor behaves as a source giving a force to others. This force can be either positive (attractive) or negative (repulsive). If two sensors are too close, they exert repulsive forces to separate each other, otherwise they exert attractive forces to draw each other. We quantify the definition of "closeness" by using the distance threshold d_{th}^{ij} for any two sensors s_i and s_j with respective sensing radius r_i and r_j (design guidelines on d_{th}^{ij} are provided in Chapter 4.1). Given k sensors (denoted as s_1, s_2, ..., s_k with sensing

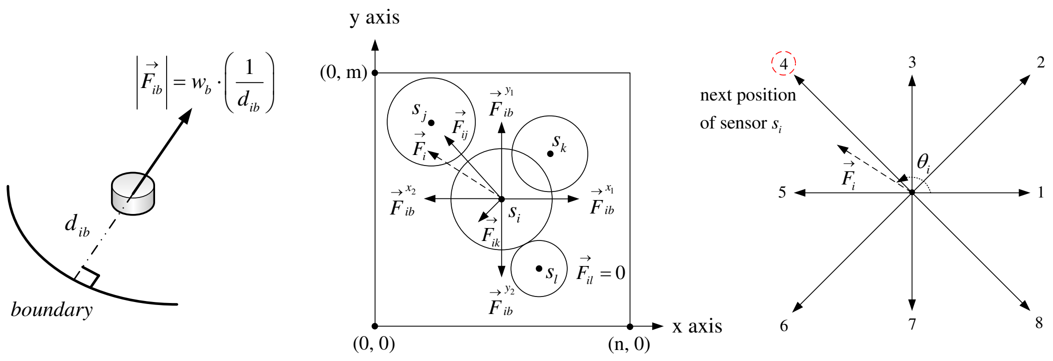


Figure 4.1: Concept of attractive, repulsive, boundary forces, and virtual movement exerted on a sensor node.

radius r_1, r_2, …, r_k, respectively) deployed in the monitoring area, for any two sensors s_i and s_j located at coordinates (x_i, y_i) and (x_j, y_j), we adopt the Euclidean distance d_{ij} to indicate how far the two sensors are spaced, where $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. As a result, if $d_{ij} > d_{th}^{ij}$, then attractive force is applied. On the other hand, repulsive force is generated if $d_{ij} < d_{th}^{ij}$. Define $\overrightarrow{F}_{ij}$ as the directed virtual force acting on s_i from s_j, now we have

$$\overrightarrow{F}_{ij} = \left\{ \begin{array}{ll} (w_a(d_{ij} - d_{th}^{ij}), \theta_{ij}) & \text{for } d_{ij} > d_{th}^{ij} \\ (0, 0) & \text{for } d_{ij}^{ij} = d_{th} \\ (w_r(d_{th}^{ij} - d_{ij}), \theta_{ij} + \pi) & \text{otherwise} \end{array} \right\}, \tag{4.1}$$

where $\theta_{ij} = \tan^{-1} \frac{(y_i - y_j)}{(x_i - x_j)}$ and w_a (w_r) represents the weight measurement for the attractive (repulsive) force (detailed design guidelines on the two weight constants are elaborated in Chapter 4.2). Take s_i in Fig. 4.1 for example, attractive force $\overrightarrow{F}_{ij}$ from s_j (to draw s_i closer) and repulsive force $\overrightarrow{F}_{ik}$ from s_k (to repel s_i) are acting simultaneously on s_i. In the case of setting distance threshold as the summation of two sensing ranges, the virtual force $\overrightarrow{F}_{il}$ from s_l equals zero (no force imposed on s_i by s_l). In addition, we incorporate the boundary force $\overrightarrow{F}_{ib}$ to quantify the virtual force acting on s_i from the monitored boundaries. By boundary forces, we can significantly reduce the unwanted coverage outside the sensing field. As depicted in Fig. 4.1, the magnitude of $\overrightarrow{F}_{ib}$ should be inversely proportional to the perpendicular distance between s_i and the boundary, and is formulated as $|\overrightarrow{F}_{ib}| = w_b(\frac{1}{d_{ib}})$, where w_b represents the weight measurement for the boundary force. In this work, we regard the boundary force as a type of repulsive force, and use the same value for w_r and w_b. In a rectangular area, boundary forces could be from the four boundaries surrounding the monitoring region. Thus $\overrightarrow{F}_{ib}$ is actually the combined force from all boundaries, where $\overrightarrow{F}_{ib} = \overrightarrow{F}_{ib}^{x_1} + \overrightarrow{F}_{ib}^{x_2} + \overrightarrow{F}_{ib}^{y_1} + \overrightarrow{F}_{ib}^{y_2}$. In Fig. 4.1, since s_i resides at the center, boundary forces from the four boundaries are equal, leading to a zero $\overrightarrow{F}_{ib}$. Considering all attractive, repulsive, and boundary forces, we have the resultant force $\overrightarrow{F}_i$ exerted on sensor s_i being defined as

$$\overrightarrow{F}_i = \sum_{j=1, j\neq i}^{k} \overrightarrow{F}_{ij} + \overrightarrow{F}_{ib}. \tag{4.2}$$

The determined resultant force $\overrightarrow{F}_i$ then guides s_i to *virtually* move to its next position. Since we adopt the discrete coordination system, the next position for s_i is defined as the closest possible grid point. As illustrated in Fig. 4.1, given the resultant moving angle θ_i, with respect to the positive x axis in counterclockwise direction, we obtain the actual motion angle $\theta_i^{'}$ by approximating $\theta_i^{'} = \frac{\pi}{4} round(\frac{\theta_i}{\pi/4})$. Consequently, sensor s_i moves to grid point 4, shown in Fig. 4.1, as its next position.

Our EVFA-B mechanism terminates when either the required sensing coverage threshold (c_{th}) is achieved or the maximum allowable virtual movements performed by each sensor ($Maxloops$) is reached.

4.1 Distance Threshold

The distance threshold effectively defines the *desired overlapping degree* of two sensors. For homogeneous sensors, the distance threshold can be made as a global constant. However, for heterogeneous sensors, the value of distance threshold should be designed on per node-pair basis to obtain a similar degree of overlapping under different sensing distances. Specifically, for two sensors with small sensing ranges, the distance threshold should be made smaller than that of two sensors with large sensing distances, in order to keep reasonably similar overlapping level for the two sensor pairs (couples). Besides sensing ranges, the design of distance threshold also depends on the sensor density. Suppose the monitoring area has size A, and the maximum area size covered by all sensors is A_s, where $A_s = \pi \sum_{i=1}^{k} r_i^2$. Define the maximum possible coverage ratio $\widetilde{a} = \frac{A_s}{A}$. Coverage

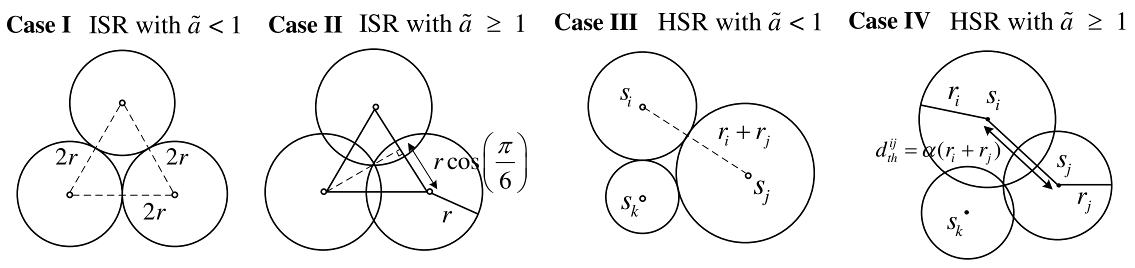

Case I ISR with $\tilde{a} < 1$
Case II ISR with $\tilde{a} \geq 1$
Case III HSR with $\tilde{a} < 1$
Case IV HSR with $\tilde{a} \geq 1$
$2r$
$2r$
$2r$
$r\cos\left(\frac{\pi}{6}\right)$
r
s_i
$r_i + r_j$
s_j
s_k
r_i
s_i
$d_{th}^{ij} = \alpha(r_i + r_j)$
s_j
r_j
s_k

Figure 4.2: Distance threshold (d_{th}^{ij}) settings for two arbitrary sensors s_i and s_j under four different environmental conditions.

ratio $\widetilde{a} < 1$ implies the total number of sensors is insufficient to fully cover the monitoring area. In this case, we cannot afford overlapping between sensors. On the other hand, coverage ratio $\widetilde{a} \geq 1$ indicates the sensor population is capable of fully covering the whole area, in which case a certain degree of overlapping is desirable to minimize the sensing holes (uncovered zones). Based on the above principles, we propose to separately design the distance threshold d_{th}^{ij} for any two sensors s_i and s_j under four environmental settings. For homogeneous sensors, we use the abbreviation ISR to reflect the fact of having Identical Sensing Radius. For heterogeneous sensors, we use HSR to represent the condition of possessing Heterogeneous Sensing Ranges. As illustrated in Fig. 4.2, **Case I** and **Case III** deal with insufficient sensor population (reflected by $\widetilde{a} < 1$) for homogeneous and heterogeneous sensors respectively, where overlapping is not desirable. As a result, the distance threshold is simply designed as the sum of two sensing ranges. In **Case II** and **Case IV**, where sensor population is sufficient to allow overlapping (due to $\widetilde{a} \geq 1$), the design of distance threshold should try to minimize the sensing holes. In **Case II**, it is easy to obtain the perfect (minimum) overlapping by setting $d_{th}^{ij} = 2r\cos(\pi/6)$, while in **Case IV**, we set $d_{th}^{ij} = \alpha(r_i + r_j)$ by introducing a system tunable factor α to control the desired overlapping degree, where $0 < \alpha < 1$. Consequently, we have the distance threshold d_{th}^{ij} being formulated in our model as

$$d_{th}^{ij} = \left\{ \begin{array}{ll} 2r & \text{for } \textbf{ISR} \text{ with } \widetilde{a} < 1 \\ 2r\cos(\frac{\pi}{6}) & \text{for } \textbf{ISR} \text{ with } \widetilde{a} \geq 1 \\ r_i + r_j & \text{for } \textbf{HSR} \text{ with } \widetilde{a} < 1 \\ \alpha(r_i + r_j) & \text{for } \textbf{HSR} \text{ with } \widetilde{a} \geq 1 \end{array} \right\}. \tag{4.3}$$

4.2 Weight Constants

For the self-deployment algorithm based on virtual forces to perform effectively in achieving high sensing coverage in a bounded $m \times n$ area, the design of weight constants w_a and w_r associated with the attractive and repulsive forces is a critical issue. Intuitively, w_r should be set much larger than w_a (as suggested in [31]), considering the relatively small number of neighboring sensors (exerting repulsive forces) compared to the large number of non-neighboring nodes out there (exerting attractive forces). However, experimental experiences reveal that arbitrary settings of a large w_r and a small w_a do not produce effective sensing coverage in many cases. In this section, we attempt to characterize the relationship between w_r and w_a by deriving a *better formulated equation* for setting the two weight constants than simply suggesting to use $w_r >> w_a$ (with arbitrary settings).

Consider an extreme node configuration shown in Fig. 4.3, where all the sensors (except for s_i and s_j) are located in one corner of the $m \times n$ sensing field. For sensor s_i, the virtual forces it receives include the repulsive force from s_j and attractive forces from all the other $(k-2)$ nodes. The magnitude of repulsive force from s_j is denoted as $|\overrightarrow{F}_i^R|$. Based on the definition of repulsive force provided in Eq. (4.1), we have $|\overrightarrow{F}_i^R| = w_r|d_{th}^{ij} - d_{ij}| = w_r\Delta$, where Δ is a small value that represents the *tolerable overlapping* between s_i and s_j. On the other hand, since the average distance between s_i and all the other $(k-2)$ nodes is approximately $(\sqrt{m^2+n^2} - \sqrt{2}r_i - \sqrt{2}r_k)$, the magnitude of total attractive forces acting on s_i is given by $|\overrightarrow{F}_i^A| = (k-2)w_a(\sqrt{m^2+n^2} - \sqrt{2}(r_i + r_k))$. Due to the relatively small

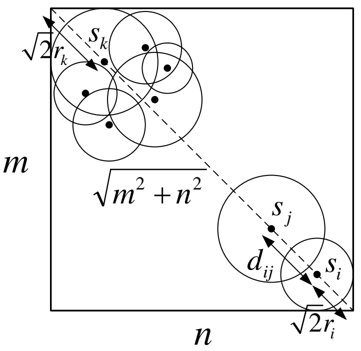


Figure 4.3: Extreme node configuration used to derive the proper $\frac{w_r}{w_a}$ ratio setting.

values of r_i and r_k compared to the area dimensions (m and n), we neglect the term $\sqrt{2}(r_i + r_k)$. Moreover, by approximating $(k-2) \approx k$, we have $|\overrightarrow{F}_i^A| = w_a k \sqrt{m^2+n^2}$. $\overrightarrow{F}_i^R$ and $\overrightarrow{F}_i^A$ are two forces that drive sensor s_i toward the opposite directions. To keep s_i in a balanced state without being drawn toward the center or pushed outside the sensing field, we adopt the equality of the two forces by making $|\overrightarrow{F}_i^R| = |\overrightarrow{F}_i^A|$. Consequently, we have

$$\frac{w_r}{w_a} = \frac{k\sqrt{m^2+n^2}}{\Delta}, \tag{4.4}$$

where m, n, and k are environmental constants, while Δ $(= |d_{th}^{ij} - d_{ij}|)$ varies with the tolerable overlapping degree of respective sensor pair (related to the sensing ranges and resultant d_{th}^{ij}). Based on the above derivations, proper choices for the weight constants can be made by setting $w_r = k\sqrt{m^2+n^2}$ and $w_a = \Delta$.

Next, we intend to further relax w_a from the dependency on sensing radius by considering setting w_a inversely proportional to the sensor population k as another alternative to the positive (attractive) weight value. In the case of having a large sensor population (with large k), the weight associated with the positive force should be made small to avoid exerting too much total attractive force on a sensor, and vice versa. To maintain a balanced force interaction, it is reasonable to relate the attractive weight measurement to the actual sensor population (parameter k). As a result, we propose another alternative

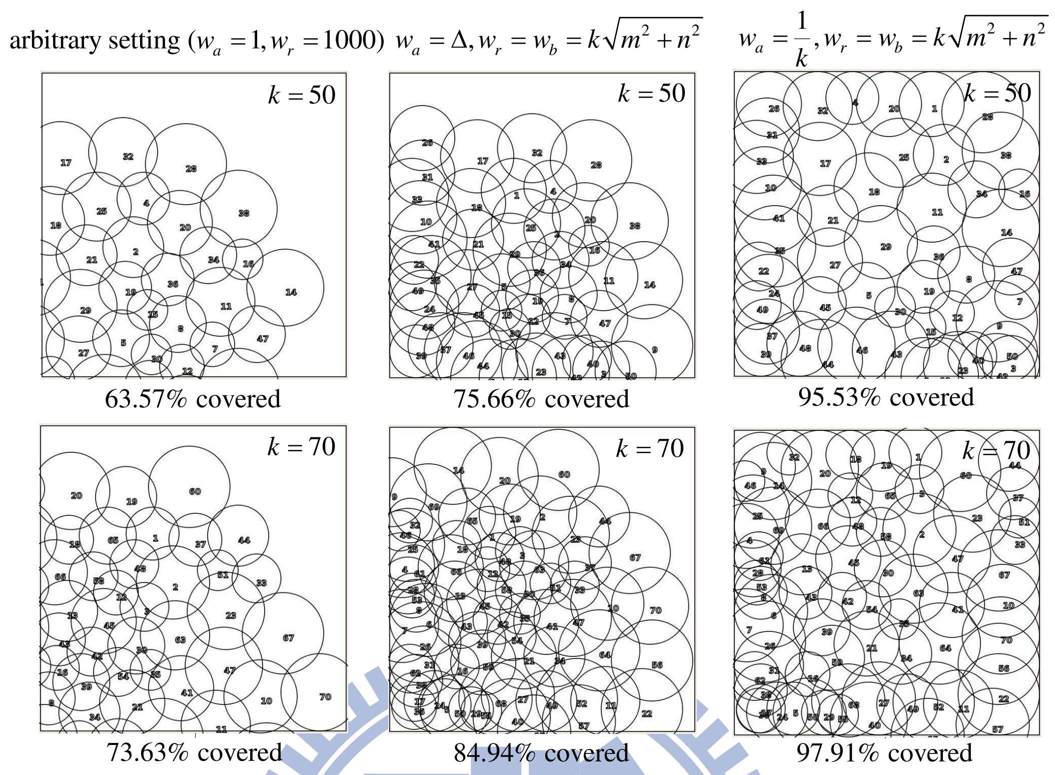

arbitrary setting ($w_a = 1, w_r = 1000$)
$w_a = \Delta, w_r = w_b = k\sqrt{m^2+n^2}$
$w_a = \frac{1}{k}, w_r = w_b = k\sqrt{m^2+n^2}$
$k = 50$
63.57% covered
75.66% covered
95.53% covered
$k = 70$
73.63% covered
84.94% covered
97.91% covered

Figure 4.4: Impact of w_a, w_r ($= w_b$) parameter settings on the coverage ratio of monitored 200×200 area (HSR with $\widetilde{\alpha} \geq 1$).

to proper weight choices by setting $w_r = k\sqrt{m^2 + n^2}$ and $w_a = \frac{1}{k}$.

In addition, since the monitored home environment is usually in a bounded area, we also incorporate the boundary forces (with weight constant w_b) in our EVFA-B mechanism. We use the same value for w_r and w_b, considering the boundary force is also a kind of repulsive force. In Fig. 4.4, we perform EVFA-B (with $Maxloops = 100$, $c_{th} = 0.95$, $\alpha = 0.9$) and experiment on two sensor populations ($k = 50$ and $k = 70$) under three different settings of w_r and w_a as discussed earlier. As we can see from the figure, arbitrary setting (though $w_r >> w_a$) without boundary forces performs poorly, while the third alternative by making w_a inversely proportional to k performs the best with the highest coverage ratio achieved. Interestingly, by setting $w_a = \frac{1}{k}$ (independent of sensing radius), we actually obtain a better sensing coverage than that by setting $w_a = \Delta$ (sensing radius dependent), which implies that *good choices for the weight constants de-*

pend on the sensor population (parameter k) and monitoring dimensions (m and n), and can be made independent of sensing radius. This implication greatly simplifies the design of weight constants when dealing with heterogeneous sensors (having varying sensing ranges). Therefore we adopt the third alternative by setting $w_r = k\sqrt{m^2+n^2}$ and $w_a = \frac{1}{k}$ in our EVFA-B mechanism thereafter.

4.3 Verification of Parameter Settings

We conduct more EVFA-B experiments ($Maxloops = 100$, $c_{th} = 0.95$) in this section to observe the combined impact of d_{th}^{ij}, w_a, w_r settings on the attainable coverage ratio. In Fig. 4.5, two d_{th}^{ij} designs are experimented (where $\bar{r} = \frac{1}{k}\sum_{i=1}^{k} r_i$, representing the average sensing radius), both with three different w_a, w_r settings. As depicted in the figure, by setting $w_a = \frac{1}{k}$ and $w_r = k\sqrt{m^2+n^2}$, we obtain the highest coverage under both d_{th}^{ij} values. Moreover, even higher coverage ratio is attainable if we make the distance threshold on per node-pair basis by setting $d_{th}^{ij} = \alpha(r_i + r_j)$. The results indicate the importance of proper parameter settings on the distance threshold (d_{th}^{ij}) and weight constants (w_a, w_r, w_b), further validating our parameter designs proposed in Chapter 4.1 and Chapter 4.2.

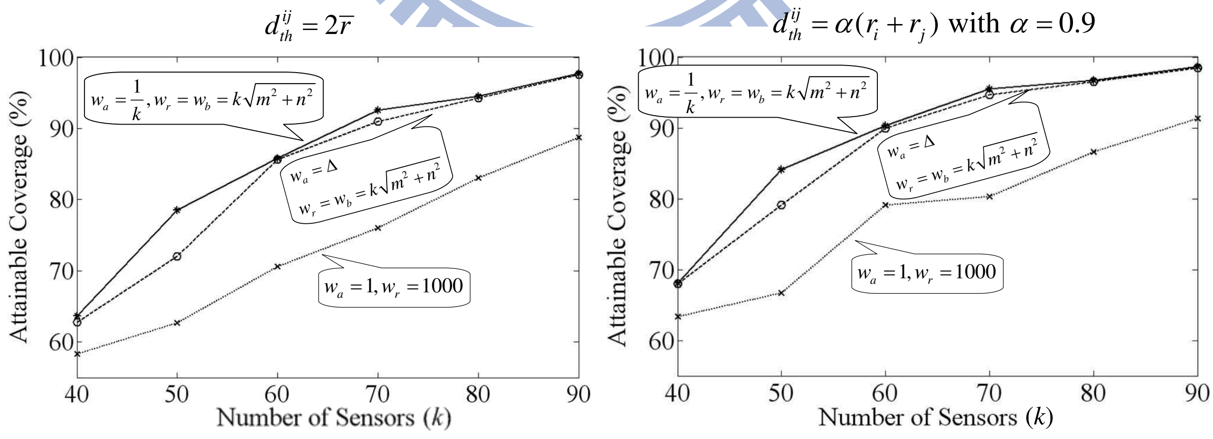


Figure 4.5: Performance justification of proper choices for $d_{th}^{ij}, w_a, w_r(w_b)$ values in our EVFA-B algorithm.

4.4 EVFA-B Algorithm Summary

Table 4.1: Summary of notations used in our EVFA-B

Notation	Description
m	Length of the monitored field
n	Width (breadth) of the monitored field
k	Total number of sensor nodes (denoted as $s_1, s_2, \cdots, s_k$ with radius $r_1, r_2, \cdots, r_k$)
(x_i, y_i)	Coordinate (position) of sensor s_i
d_{th}^{ij}	Distance threshold for two arbitrary sensors s_i and s_j ($j \neq i$)
w_a	Tunable measure weight for the attractive force
$w_r(w_b)$	Tunable measure weight for the repulsive force (boundary force)
$\overrightarrow{F_i}$	Resultant force exerted on sensor s_i (attractive, repulsive, boundary forces considered)
$Maxloops$	Maximum number of virtual movements performed by each sensor
c_{th}	Desired coverage ratio threshold

Algorithm 1 Enhanced Virtual Forces Algorithm with Boundary Forces (EVFA-B)

```
 1: set $loops = 0$;
 2: set $c_{now} = c_{init}$;  // initial coverage ratio
 3: while ($loops < Maxloops$) && ($c_{now} < c_{th}$) do
 4:     for each sensor $s_i \in \{s_1, s_2, ..., s_k\}$ do
 5:         compute $\overrightarrow{F}_i = \sum_{j \neq i, j=1}^{k} \overrightarrow{F}_{ij} + \overrightarrow{F}_{ib}$;
 6:     end for
 7:     perform virtual movements;  // all sensors virtually move to their next positions
 8:     update coverage ratio $c_{now}$;
 9:     set $loops = loops + 1$;
10: end while
```

Table 4.1 summarizes the notations used in the EVFA-B mechanism, and Algorithm 1 provides the pseudocode for EVFA-B operations. Note that in the end of each loop, every sensor performs *virtual movement* without physically moving to the new position. Physical movements are conducted once the EVFA-B process terminates (either c_{th} or $Maxloops$ has been reached), and this is when our collision-free path planning (CFPP) algorithm (detailed in Chapter 5) comes into play.

Chapter 5

Collision-Free Path Planning (CFPP)

In practical deployment, a collision-free moving path scheduling is essential, so that mobile sensors can reach their destinations without colliding with each other. However, the scheduling strategy is non-trivial for various collision cases need be systematically classified and handled/resolved in different ways. In this work, we assume the sensor volume is neglected and regarded as a *moving point* on a 2D plane, while every moving path (performed by a sensor) regarded as a *line*. Suppose no two moving paths share the same line (i.e., no path lies in the sub-path of another). We identify the collision cases based on the following geometric theorem.

Theorem 1. *With respect to the line $ax + by + c = 0$ on a 2D plane, points $Q_1(x_1, y_1)$ and $Q_2(x_2, y_2)$ fall in the same side if $(ax_1 + by_1 + c)(ax_2 + by_2 + c) > 0$, in different sides if $(ax_1 + by_1 + c)(ax_2 + by_2 + c) < 0$, while one or both reside(s) exactly on the line if $(ax_1 + by_1 + c)(ax_2 + by_2 + c) = 0$.*

For an arbitrary sensor s_i departing from point p_i (with coordinate $(x_i,\ y_i)$) to point $p_i^{'}$ (with coordinate $(x_i^{'},\ y_i^{'})$), the moving path can be formulated as a line, denoted as

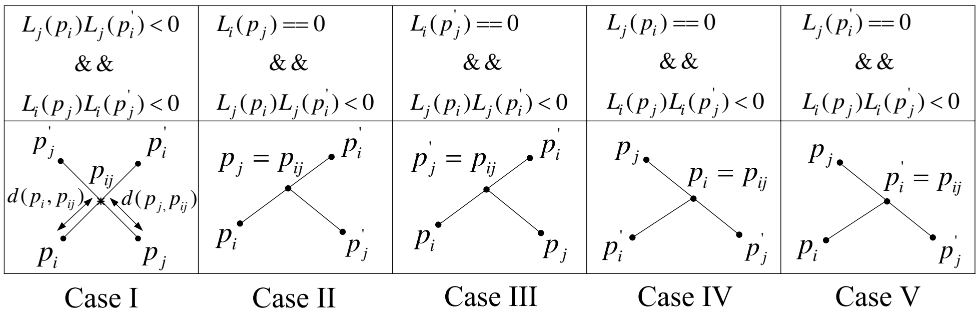

$L_j(p_i)L_j(p_i^{'})<0$ && $L_i(p_j)L_i(p_j^{'})<0$
$L_i(p_j)==0$ && $L_j(p_i)L_j(p_i^{'})<0$
$L_i(p_j^{'})==0$ && $L_j(p_i)L_j(p_i^{'})<0$
$L_j(p_i)==0$ && $L_i(p_j)L_i(p_j^{'})<0$
$L_j(p_i^{'})==0$ && $L_i(p_j)L_i(p_j^{'})<0$
$p_j^{'}$ $p_i^{'}$ p_{ij} $d(p_i,p_{ij})$ $d(p_j,p_{ij})$ p_i p_j
$p_j = p_{ij}$ $p_i^{'}$ p_i $p_j^{'}$
$p_j^{'} = p_{ij}$ $p_i^{'}$ p_i p_j
p_j $p_i = p_{ij}$ $p_i^{'}$ $p_j^{'}$
p_j $p_i^{'} = p_{ij}$ p_i $p_j^{'}$
Case I
Case II
Case III
Case IV
Case V

Figure 5.1: Possible intersection (collision) cases generated by moving paths of any two sensors s_i and s_j, where p_i (p_j) denotes the original position of s_i (s_j) and $p_i^{'}$ ($p_j^{'}$) indicates the physical movement destination for sensor s_i (s_j).

L_i. Similarly, the moving path of another sensor s_j is given as L_j. Define p_{ij} as the *intersection point* of lines L_i and L_j, which can be easily obtained by solving the two line equations. According to Theorem 1, we can now classify five possible intersection (collision) cases for any two sensors s_i and s_j, as illustrated in Fig. 5.1, where $d(p_i, p_{ij})$ and $d(p_j, p_{ij})$ represent the Euclidean distances from p_i to p_{ij} and from p_j to p_{ij}. **Case I** shows the case in which points p_i and $p_i^{'}$ fall in different sides of line L_j, while points p_j and $p_j^{'}$ fall in different sides of line L_i as well. In **Case II**, the departure point p_j of sensor s_j gets in the way of the moving path of s_i, while in **Case IV**, on the contrary, the departure point p_i of sensor s_i blocks the moving path of s_j. **Case III** draws the condition in which the destination point $p_j^{'}$ of sensor s_j lies on the moving path of s_i, while **Case V**, on the opposite side, displays the condition that destination point $p_i^{'}$ of sensor s_i falls on the moving path of s_j.

5.1 Path Planning Strategy

Given the five potential collision (intersection) cases caused by any two moving paths, we establish colliding set C_i, which includes all sensors whose moving paths intersect with that of s_i, for each sensor. Instead of performing one-time physical movements, we propose to use *batched movements* such that the scheduling complexity can be reduced at the expenses of increased moving latency. Define $order_i$ as the cardinality of set C_i ($order_i = |C_i|$) for sensor s_i, indicating its moving order. We start from performing movements for sensors with the least $order$ value. All sensors with the currently least (smallest) $order$ value are contained in set M_{min_order}. Intuitively, sensors with $order$ value of zero can move simultaneously since no other sensors pose potential colliding sources to them. For any sensor s_i with non-zero $order_i$ value, potential colliding conditions (on per node-pair basis) caused by all members in its C_i set should be analyzed and handled case by case. Specifically, all sensors are divided into moving groups (batches) based on their $order$ values and processed round by round (batch by batch). Sensors in set M_{min_order} are evaluated in the same round. The evaluation and processing details will be provided later in this section. After the evaluations, a subset of M_{min_order} (or probably the whole M_{min_order} set) is determined and all sensors included in the subset are allowed to move simultaneously in the current round. For sensor s_i that has been evaluated and permitted to move, the $tflag_i$ is set $true$, indicating its moving intention. Once the physical movement has been successfully performed by sensor s_i, moving flag $mflag_i$ is set $true$ and s_i is removed from the consideration list. All $order$ values for the remaining sensors (physical movements not performed yet) should be refreshed, and the batched scheduling procedure starts over accordingly.

Now we detail on the evaluation procedures for determining a set of movable sensors in a single round (batch). Based on the idea of batched movements, we regard all sensors with the currently minimum $order$ value as a potential moving batch and include them in set M_{min_order}. We then analyze all members in set M_{min_order} one by one to determine their moving possibilities. In our design, we start the evaluation from sensor with the smallest ID, say s_1, and identify all possible collision cases caused by members in its colliding set C_1. For any two sensors s_i and s_j with moving orders $order_i$ and $order_j$, the previously five collision cases can be further classified into ten cases according to the relationship of $order_i$ and $order_j$. Suppose $s_i \in M_{min\_order}$, $s_j \in C_i$, and $order_i = order_j$, we term the five collision cases as **Case S-I**, **Case S-II**, **Case S-III**, **Case S-IV**, and **Case S-V**, where 'S' indicates that sensors s_i and s_j are potentially scheduled to move in the "same" round due to equal *order* value. On the other hand, if $order_i < order_j$ (note that $order_i > order_j$ is not possible since $s_i \in M_{min\_order}$), we define another five collision cases as **Case D-I**, **Case D-II**, **Case D-III**, **Case D-IV**, and **Case D-V**, where 'D' means s_i and s_j are potentially scheduled to move in "different" rounds due to their unequal *order* values. In each potential collision case, on detecting a colliding possibility, s_i tries to resolve the collision by adjusting/prolonging the waiting time T_j or increasing the moving speed V_j of sensor s_j. Originally all waiting times are set to zero, and moving speeds all set at a constant velocity V. If the adjustment (on either waiting time or moving speed) is successful, the colliding possibility is eliminated and s_i moves on to evaluate collision cases with other members in C_i. To avoid repeated adjustments on a single sensor, in our design, each sensor is allowed to be adjusted (either on waiting time or moving velocity) *once*. In addition, s_i itself cannot be adjusted by other sensors in set M_{min_order} that are evaluated *after* it, if s_i is indeed scheduled to move in the current round. We keep track of the adjustment possibility for sensor s_i by the $dirty_i$ bit, implying adjustable if set $false$ and not adjustable if set *true*. When s_i intends to resolve a collision by adjusting another sensor with *dirty* bit set *true*, the adjustment is prohibited and s_i is not allowed to move in the current round ($tflag_i$ set to $false$), since the collision remains. Only when all members in C_i with various colliding possibilities are all resolved can sensor s_i be included into the movable set and perform physical movement. Upon receiving the moving instruction from the clusterhead, s_i waits for T_i (possibly adjusted) and then moves with speed V_i (possibly adjusted). In our route scheduling strategy, we try to include *as many sensors as possible* to move simultaneously in the same round (batch).

For each of the ten collision cases identified, we define corresponding actions (**Action D-I**, **Action D-II**, $\cdots$, **Action S-I**, **Action S-II**, $\cdots$) to evaluate respective case and perform necessary adjustments. If colliding possibility remains due to unsuccessful adjustment, physical movement by sensor s_i is not allowed and should be deferred. Thus we additionally define **Action Deferred** to perform corresponding operations. Note that in **Case D-I**, **Case D-III**, and **Case D-IV**, no action is needed since s_i and s_j are scheduled in different rounds (no collision is likely to happen in the three cases despite intersection exists between the two moving paths). For the rest of seven cases, we describe the evaluation principles exercised by respective action as follows (detailed operations are available in Algorithm 2, Chapter 5.2).

Action D-II In this case, since s_j gets in the way of s_i's moving path, the clusterhead instructs s_j to slightly adjust its location along line $\overrightarrow{p_j p_j^{'}}$ to avoid collision. Assume the location adjustment is small enough to have no effect on other moving paths.

Action D-V Sensor s_i is not allowed to move, for its destination point $p_i^{'}$ will block the moving path of s_j in a later round. In this case, the moving order of s_i should be set to be larger than that of s_j ($order_i = order_j + 1$) to postpone s_i's physical movement after s_j. In addition, a fix_order_i flag should be set *true*, indicating no updates on $order_i$ will be performed in later rounds to ensure the delayed movement after s_j, and then **Action Deferred** is invoked for s_i.

Action S-I Define the traveling time from p_i to the intersection point p_{ij} as $t_{p_i \to p_{ij}}$ (obtained from available $d(p_i, p_{ij})$ and V_i), the clusterhead evaluates if $T_i + t_{p_i \to p_{ij}} = T_j + t_{p_j \to p_{ij}}$, where T_i and T_j are the waiting times of s_i and s_j as defined earlier. If equality holds, a collision at the intersection is expected, and the waiting time T_j of s_j should be increased by a small amount of Δt to avoid the collision. However, in case s_j has already been processed with $dirty_j$ set *true*, the adjustment is prohibited and s_i is not allowed to move in the current round. Consequently, moving order of s_i is increased

$(order_i = order_i + 1)$ and **Action Deferred** is invoked for s_i.

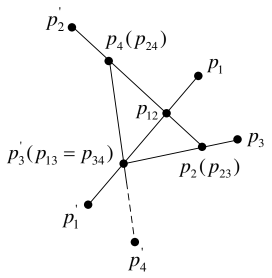
Action S-II If s_i reaches the intersection point p_{ij} no later than s_j's departure time, the clusterhead should instruct s_j to slightly adjust its location along line $\overrightarrow{p_j p_j^{'}}$ to avoid collision.

Action S-III If s_j reaches the intersection point p_{ij} no later than s_i, the destination point $p_j^{'}$ of s_j will block the moving path of s_i. In this case, the clusterhead should instruct s_j to increase its waiting time T_j by setting $T_j = T_i + (t_{p_i \to p_{ij}} - t_{p_j \to p_{ij}}) + \Delta t$ to ensure the delayed arrival of s_j at p_{ij} $(p_j^{'})$. If the adjustment of T_j is not successful due to a *true* flag of $dirty_j$, then s_j is not allowed to move in the current round. Consequently, moving order of s_j is increased $(order_j = order_j + 1)$ and **Action Deferred** is invoked for s_j.

Action S-IV If s_j reaches the intersection point p_{ij} no later than s_i's departure time, the clusterhead should increase the waiting time of s_j by setting $T_j = T_i - t_{p_j \to p_{ij}} + \Delta t$. In case the adjustment is not allowed due to a *true* value of $dirty_j$, the clusterhead instructs s_i to slightly adjust its location along line $\overrightarrow{p_i p_i^{'}}$ to avoid collision.

Action S-V If s_i reaches the intersection point p_{ij} no later than s_j, the destination point $p_i^{'}$ of s_i will block the moving path of s_j. In this case, the clusterhead should instruct s_j to increase its moving speed V_j by setting $V_j = \frac{V_i \cdot d(p_j, p_{ij})}{d(p_i, p_{ij}) + V_i(T_i - T_j)} + \Delta v$, where Δv is a small amount of speed increment to ensure s_j's earlier arrival at p_{ij} $(p_i^{'})$ than s_i. However, if the adjusted V_j is larger than the maximum possible moving speed V_{max} or the adjustment of V_j is prohibited due to a *true* value of $dirty_j$, then s_i is not allowed to move in the current round. Moving order of s_i is increased $(order_i = order_i + 1)$ and **Action Deferred** is invoked for s_i.

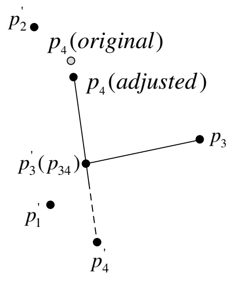
Action Deferred Since s_i (s_j) is not allowed to move in the current round, $tflag_i$ $(tflag_j)$ is set $false$. In addition, the clusterhead should confirm if this not-moving decision leads to moving path blocking of any sensor in M_{min_order} set that is already allowed to move in the current round (with $tflag$ set *true*), and do necessary slight



(a) Before moving

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1	$s_2(\text{S}-\text{I}),\ s_3(\text{S}-\text{III}),\ s_4(\text{D}-\text{I})$	3	0	1	1	0
s_2	$s_1(\text{S}-\text{I}),\ s_3(\text{S}-\text{IV}),\ s_4(\text{D}-\text{II})$	3	0	1	1	0
s_3	$s_1(\text{S}-\text{V}),\ s_2(\text{S}-\text{II}),\ s_4(\text{D}-\text{V})$	3	0	1	0	0
s_4	$s_1(\text{D}-\text{I}),\ s_2(\text{D}-\text{IV}),\ s_3(\text{D}-\text{III})\cdots$	5	0	0	0	0

$\text{M}_{\text{min\_order}} = \{s_1,\ s_2,\ s_3\}$



s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1						1
s_2						1
s_3	$s_4(\text{D}-\text{V})$	6	1	0	0	0
s_4	$s_3(\text{D}-\text{III})\cdots$	3	0	0	0	0

Moving set = $\{s_1,\ s_2\}$

(b) After moving

Figure 5.2: Every sensor s_i in the potential moving set M_{min_order} should be analyzed by identifying its intersection (collision) relationship with each member in C_i, in which intersection cases D-II, D-IV, S-I, S-II, S-III, S-IV, and S-V require further consideration/processing, before including s_i into the moving set (allowed to move in the current round).

location adjustment to resolve the blocking.

Fig. 5.2 illustrates a snapshot of the CFPP operations. Note that s_4 has more intersections with other sensors, which are not shown in the figure (omitted for brevity). In the current round, potential moving set M_{min_order} includes s_1, s_2, and s_3, all having the currently smallest *order* value of 3. For s_1, colliding conditions caused by all members in C_1 are analyzed and handled case by case. In this example, since s_1 and s_2 are evaluated to reach intersection p_{12} simultaneously, the clusterhead adjusts the waiting time of s_2 by setting $T_2 = T_2 + \Delta t$ to resolve the collision. Next, since s_3 is found to reach intersection p_{13} earlier than s_1, blocking s_1's moving path, the clusterhead instructs s_3 to increase its waiting time by setting $T_3 = T_1 + (t_{p_1 \to p_{13}} - t_{p_3 \to p_{13}}) + \Delta t$. As to s_4 (scheduled to move in a later round), no action is required since no collision is likely to happen between s_1 and s_4. Consequently, the clusterhead includes s_1 into the moving set. Similar operations apply to s_2. In our example, s_2 has no colliding possibilities with s_1 and s_3. However, since the departure location p_4 of s_4 blocks s_2's moving path, the clusterhead instructs s_4 to slightly move from p_4 (original) to p_4 (adjusted), as shown in Fig. 5.2 (b). As a result, s_2 is also included into the moving set. For s_3, in our example, both s_1 and s_2 do not pose colliding sources to s_3. Unfortunately, since the destination point $p_3^{'}$ of s_3 will block the moving path of s_4 in a future round, s_3 is not allowed to move before s_4 (not included into the moving set), and $order_3$ should be updated to 6 ($order_4 + 1$) with fix_order_3 set *true*. After the evaluations, sensors included in the moving set (i.e., s_1 and s_2) perform physical movements simultaneously, and $order_4$ and set C_4 are updated accordingly.

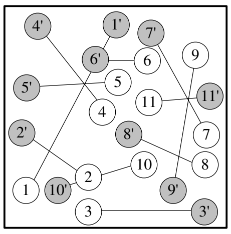
5.2 CFPP Algorithm Summary

Table 5.1: Summary of notations used in the CFPP algorithm

Notation	**Description**
C_i	Set of potential colliding sensors against s_i
$order_i$	Moving order of s_i, where $order_i = \|C_i\|$
fix_order_i	Indicates the $order$ value of s_i is henceforth fixed
$dirty_i$	Indicates whether s_i has been processed in the current round
$tflag_i$	Indicates whether s_i is allowed to move in the current round
$mflag_i$	Indicates whether s_i has moved from p_i to $p_i^{'}$
M_{min_order}	Set of sensors with the minimum $order$ value in the current round

Table 5.1 summarizes the notations used in CFPP, and Algorithm 2 provides the pseudocode for CFPP operations. In addition, a running example illustrating the CFPP route scheduling procedures is available in Fig. 5.3. Note that in Round 3 of this example, s_9 is excluded from the moving set due to a unsuccessful adjustment of s_{11}'s waiting time (since T_{11} has been adjusted by the clusterhead to resolve collision with s_7 and can only be adjusted *once* according to the scheduling principles adopted by CFPP). After the clusterhead decides that s_9 is not allowed to move in the current round, s_9 no longer poses as a colliding source to s_{11}. Consequently, s_{11} can be included into the moving set

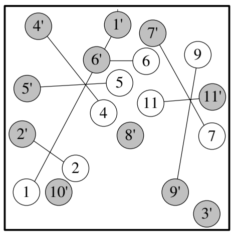
Initial



s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1	s_2, s_4, s_5, s_6	4	0	0	0	0
s_2	s_1, s_{10}	2	0	0	0	0
s_3		0	0	1	1	0
s_4	s_1, s_5	2	0	0	0	0
s_5	s_1, s_4	2	0	0	0	0
s_6	s_1	1	0	1	0	0
s_7	s_9, s_{11}	2	0	0	0	0
s_8	s_9	1	0	1	1	0
s_9	s_7, s_8, s_{11}	3	0	0	0	0
s_{10}	s_2	1	0	1	1	0
s_{11}	s_7, s_9	2	0	0	0	0

$\mathrm{M}_{\mathrm{min\_order}} = \{s_3, s_6, s_8, s_{10}\}$

Round 1

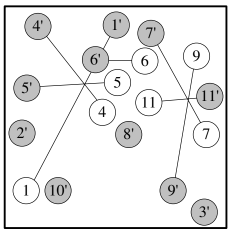


Moving set = $\{s_3, s_8, s_{10}\}$

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1	s_2, s_4, s_5, s_6	4	0	0	0	0
s_2	s_1	1	0	1	1	0
s_3						1
s_4	s_1, s_5	2	0	0	0	0
s_5	s_1, s_4	2	0	0	0	0
s_6	s_1	5	1	0	0	0
s_7	s_9, s_{11}	2	0	0	0	0
s_8						1
s_9	s_7, s_{11}	2	0	0	0	0
s_{10}						1
s_{11}	s_7, s_9	2	0	0	0	0

$\mathrm{M}_{\mathrm{min\_order}} = \{s_2\}$

Round 2

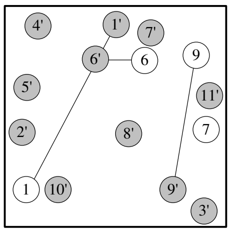


Moving set = $\{s_2\}$

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1	s_4, s_5, s_6	3	0	0	0	0
s_2						1
s_3						1
s_4	s_1, s_5	2	0	1	1	0
s_5	s_1, s_4	2	0	1	1	0
s_6	s_1	5	1	0	0	0
s_7	s_9, s_{11}	2	0	1	1	0
s_8						1
s_9	s_7, s_{11}	2	0	1	0	0
s_{10}						1
s_{11}	s_7, s_9	2	0	1	1	0

$\mathrm{M}_{\mathrm{min\_order}} = \{s_4, s_5, s_7, s_9, s_{11}\}$

Round 3

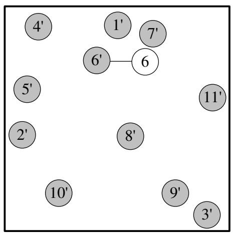


Moving set = $\{s_4, s_5, s_7, s_{11}\}$

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1	s_6	1	0	1	1	0
s_2						1
s_3						1
s_4						1
s_5						1
s_6	s_1	5	1	0	0	0
s_7						1
s_8						1
s_9		0	0	1	1	0
s_{10}						1
s_{11}						1

$\mathrm{M}_{\mathrm{min\_order}} = \{s_1, s_9\}$

Round 4

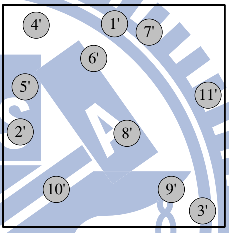


Moving set = $\{s_1, s_9\}$

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1						1
s_2						1
s_3						1
s_4						1
s_5						1
s_6		5	1	0	0	0
s_7						1
s_8						1
s_9						1
s_{10}						1
s_{11}						1

$\mathrm{M}_{\mathrm{min\_order}} = \{s_6\}$

Round 5 (Completed)



Moving set = $\{s_6\}$

s_i	C_i	$order_i$	fix_order_i	$dirty_i$	$tflag_i$	$mflag_i$
s_1						1
s_2						1
s_3						1
s_4						1
s_5						1
s_6						1
s_7						1
s_8						1
s_9						1
s_{10}						1
s_{11}						1

Figure 5.3: Example illustrating the operations of our proposed CFPP algorithm for sensor physical movements.

by the clusterhead.

In the CFPP strategy, we propose batched movements to successfully resolve moving collisions between sensors at the cost of global deployment latency. While most existing self-deployment works do not handle this collision problem, our proposed CFPP strategy is essential in practical deployment, and we believe the disadvantage of increased deployment time can be effectively reduced by the local recovery capability provided by our SSOA mechanism (detailed in Chapter 6), which leads to infrequent global redeployments.

Algorithm 2 Collision-free Path Planning (CFPP)

```
include all sensors in set $S$;
establish set $C_i$ for $\forall s_i \in S$ ; // $i = 1, \cdots, k$
evaluate $order_i$ for $\forall s_i \in S$;
clear $fix\_order_i$, $dirty_i$, $tflag_i$, $mflag_i$ for $\forall s_i \in S$; // all set to $false$
while ($S$ !empty) do
    re-establish set $C_i$ for $\forall s_i \in S$;
    re-evaluate $order_i$ for $\forall s_i \in S$ with $fix\_order_i == false$;
    reset $T_i = 0, V_i = V$ for $\forall s_i \in S$;
    include all $s_i$ with the minimum $order_i$ value into the $M_{min\_order}$ set;
    for (each $s_i \in M_{min\_order}$) do
        set $dirty_i = true$; set $tflag_i = true$;
        for (each $s_j \in C_i$) do
            classify the intersection (collision) $case$ for $s_i$ and $s_j$;
            switch ($case$)
                Case D-II: do Action D-II;
                Case D-V: do Action D-V;
                Case S-I: do Action S-I;
                Case S-II: do Action S-II;
                Case S-III: do Action S-III;
                Case S-IV: do Action S-IV;
                Case S-V: do Action S-V;
        end for
    end for
    perform simultaneous physical movements for $\forall s_i$ with $tflag_i == true$;
    set $mflag_i = true$ for such sensor $s_i$; // indicating physical movement performed
    remove all $s_i$ with $mflag_i == true$ from sensors set $S$;
end while
procedure Action D-II
    slightly adjust location of $s_j$ from $p_j$ (original) to $p_j$ (adjusted);
procedure Action D-V
    set $order_i = order_j + 1$; set $fix\_order_i = true$; invoke Action Deferred ($s_i$);
procedure Action S-I
    if $T_i + t_{p_i \to p_{ij}} = T_j + t_{p_j \to p_{ij}}$ then
        if $dirty_j == false$ then set $T_j = T_j + \Delta t$; $dirty_j = true$;
        else set $order_i = order_i + 1$; invoke Action Deferred ($s_i$);
procedure Action S-II
    if $T_i + t_{p_i \to p_{ij}} \leq T_j$ then slightly adjust location of $s_j$ from $p_j$ (original) to $p_j$ (adjusted);
procedure Action S-III
    if $T_i + t_{p_i \to p_{ij}} \geq T_j + t_{p_j \to p_{ij}}$ then
        if $dirty_j == false$ then set $T_j = T_i + (t_{p_i \to p_{ij}} - t_{p_j \to p_{ij}}) + \Delta t$; set $dirty_j = true$;
        else set $order_j = order_j + 1$; invoke Action Deferred ($s_j$);
procedure Action S-IV
    if $T_i \geq T_j + t_{p_j \to p_{ij}}$ then
        if $dirty_j == false$ then set $T_j = T_i - t_{p_j \to p_{ij}} + \Delta t$; set $dirty_j = true$;
        else slightly adjust location of $s_i$ from $p_i$ (original) to $p_i$ (adjusted);
procedure Action S-V
    if $T_i + t_{p_i \to p_{ij}} \leq T_j + t_{p_j \to p_{ij}}$ then
        if $dirty_j == false$ then set $V_j = \frac{V_i d(p_j,p_{ij})}{d(p_i,p_{ij}) + V_i(T_i - T_j)} + \Delta v$; set $dirty_j = true$;
            if $V_j > V_{max}$ then set $order_i = order_i + 1$; invoke Action Deferred ($s_i$);
        else set $order_i = order_i + 1$; invoke Action Deferred ($s_i$);
procedure Action Deferred ($s_i$)
    set $tflag_i = false$; do necessary slight adjustment of $s_i$'s departure location to resolve
    moving path blocking possibly caused by this not-moving decision;
```

Chapter 6

Sensor Self-Organizing Algorithm (SSOA)

Wireless sensors are inherently unreliable. Due to sensor power depletions or unexpected failures over time, the decreased sensing coverage deteriorates the event detection capability of a WSN. To preserve the required sensing coverage, one alternative is to perform EVFA-B (presented in Chapter 4) periodically for global redeployments. However, such constant global redeployment is costly in terms of communication overhead and consumed moving energy, and should be kept infrequent. Therefore, we propose the sensor self-organizing algorithm (SSOA) to firstly repair the sensing void (uncovered area caused by some broken sensor) by locally repositioning sensors around the sensing hole. Two issues need be addressed to realize this local recovery: selection of repairing sensors (Chapter 6.1) and physical movements performed by the selected sensors (Chapter 6.2). In case the local repairing is unable to recover the required sensing (detection) capability, SSOA then invokes EVFA-B to globally redeploy sensors.

6.1 Local Selection of Rescue Sensors

The first challenge of accomplishing partial repair is to locally select the rescue sensors around the sensing hole. Given a sensing hole caused by some broken sensor (s_{dead}), all active sensors nearby (not necessarily the immediate neighbors of s_{dead}) can be potential candidates to perform the local repair. Theoretically, every combination of rescue sensor candidates along with various moving strategies should be examined to obtain the most desirable coverage improvement. However, this approach is intractable, and not implementable. Therefore, we limit the search of rescue sensors to the neighboring nodes of s_{dead}, defined as set N_{dead}. Our objective is to select a subset R_{dead} of local rescue sensors from N_{dead} (i.e., $R_{dead} \subseteq N_{dead}$) for repairing the sensing hole.

In order to evaluate the recovering capability of each sensor $s_i \in N_{dead}$, we try to quantify the *overlapping degree* possessed by each sensor, and associate an overlapping weight w_i with sensor s_i. As shown in Fig. 6.1, for any two sensors s_i and s_j with sensing radius r_i and r_j respectively, the overlapping degree w_{ij} is defined as the overlapped area between the two sensors, and can be easily obtained as $w_{ij} = r_i^2\theta_i + r_j^2\theta_j - d_{ij}r_i \sin\theta_i$. Then the overlapping degree w_i can be approximated by summing up overlapping weights contributed from all neighbors of s_i, thus we have $w_i = \sum_{s_j \in N_i} w_{ij}$ (recall that N_i represents the neighbor set of s_i). However, considering the existing sensing hole(s) around s_i, the overlapping degree should be adjusted by deducting the uncovered area(s) from w_i to reflect this fact. Thus we have the modified $w_i = \sum_{s_j \in N_i} w_{ij} - w_h$, where w_h represents the area size collectively contributed by sensing hole(s) around s_i. The estimation of w_h can be obtained by some existing geometric calculations [19]. As a result, the quantified overlapping degree w_i can be either *positive* or *negative*, reflecting the recovering capability of rescue sensor candidate s_i ($s_i \in N_{dead}$).

Intuitively, higher overlapping degree w_i implies better recovering ability of a sensor candidate s_i. Define the total overlapping degree of a selected rescue sensors set R_{dead}

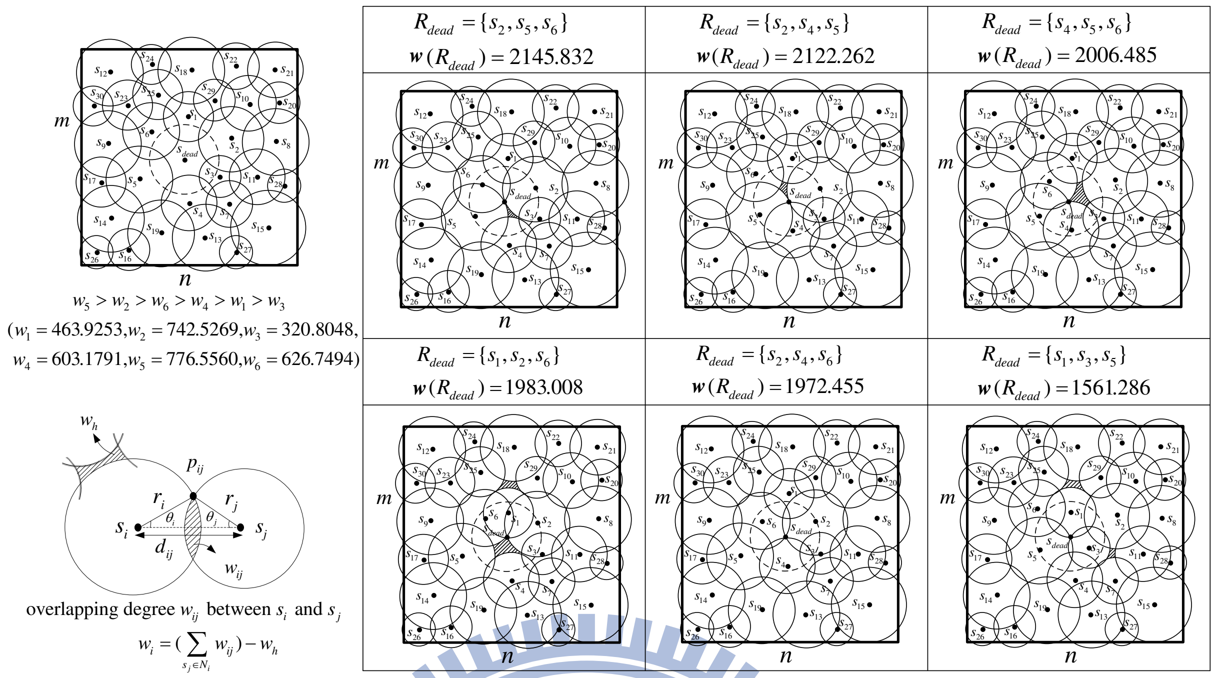

$w_5 > w_2 > w_6 > w_4 > w_1 > w_3$
$(w_1 = 463.9253, w_2 = 742.5269, w_3 = 320.8048,$
$w_4 = 603.1791, w_5 = 776.5560, w_6 = 626.7494)$
overlapping degree w_{ij} between s_i and s_j
$w_i = (\sum_{s_j \in N_i} w_{ij}) - w_h$
$R_{dead} = \{s_2, s_5, s_6\}$ $w(R_{dead}) = 2145.832$
$R_{dead} = \{s_2, s_4, s_5\}$ $w(R_{dead}) = 2122.262$
$R_{dead} = \{s_4, s_5, s_6\}$ $w(R_{dead}) = 2006.485$
$R_{dead} = \{s_1, s_2, s_6\}$ $w(R_{dead}) = 1983.008$
$R_{dead} = \{s_2, s_4, s_6\}$ $w(R_{dead}) = 1972.455$
$R_{dead} = \{s_1, s_3, s_5\}$ $w(R_{dead}) = 1561.286$

Figure 6.1: Experiments on possible selections of rescue sensors set R_{dead} to locally recover the sensing void caused by faulty sensor s_{dead}.

as $\boldsymbol{w}(R_{dead})$, where $\boldsymbol{w}(R_{dead}) = \sum_{s_i \in R_{dead}} w_i$. A selected rescue sensors set R_{dead} with a higher $\boldsymbol{w}(R_{dead})$ is expected to achieve better coverage improvement. However, as illustrated in Fig. 6.1, our experiments on 30 heterogeneous sensors deployed in a 125×125 area reveal that the highest $\boldsymbol{w}(R_{dead})$ by selecting $R_{dead} = \{s_2, s_5, s_6\}$ does not produce the best coverage performance. On the other hand, also containing three rescue sensors, the set $R_{dead} = \{s_2, s_4, s_6\}$ with the fifth highest $\boldsymbol{w}(R_{dead})$ leads to the best coverage improvement among the six cases. From extensive experiments conducted (not shown in the thesis), we observe that selecting adjacent sensors (though with high overlapping degrees) to move simultaneously usually leads to unnecessary overlapping and cannot effectively cover the sensing hole. On the contrary, selecting non-adjacent sensors, such as $R_{dead} = \{s_2, s_4, s_6\}$, to cooperatively repair the sensing void generally produces effective coverage. The results suggest that the impact of locations of selected rescue sensors (non-adjacent nodes preferred) seems to be more pronounced than that of overlapping

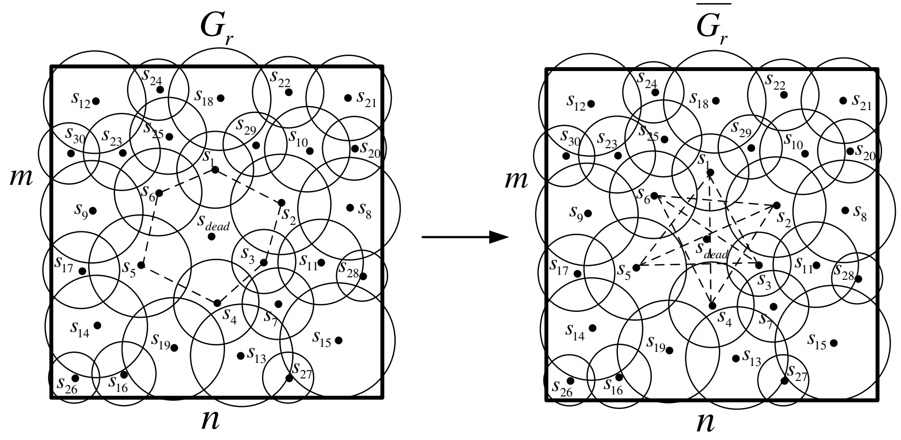


Figure 6.2: Construction of graph $\overline{G_r}$ for our rescue sensors selection problem (RSSP).

degrees. Nonetheless, overlapping degree is still important, for the selection of $R_{dead} = \{s_1, s_3, s_5\}$, as shown in Fig. 6.1, results in imperfect coverage due to its insufficient $\boldsymbol{w}(R_{dead})$. Consequently, in this work, *we propose to select a rescue sensors set R_{dead} that contains non-adjacent sensors in N_{dead} with the highest $\boldsymbol{w}(R_{dead})$ value.*

Given both positive and negative overlapping weights, however, the R_{dead} combinations of non-adjacent nodes selected from N_{dead} can be many (with various R_{dead} set sizes). Specifically, the best R_{dead} (including non-adjacent sensors) with the highest $\boldsymbol{w}(R_{dead})$ that we intend to obtain may contain $1, 2, \ldots,$ and up to $\lfloor \frac{|N_{dead}|}{2} \rfloor$ nodes. In other words, $C_1^{|N_{dead}|} + C_2^{|N_{dead}|} + \cdots + C_{\lfloor \frac{|N_{dead}|}{2} \rfloor}^{|N_{dead}|}$ candidate combinations should be tried out to obtain the best set containing only non-adjacent nodes and having the highest overlapping degree. Due to the inefficient computational complexity required by the above selection approach, we try to further reduce the candidate space. Suppose θ_i denotes the angle produced by line segment $\overline{s_i s_{dead}}$ ($s_i \in N_{dead}$) with respect to the positive x axis in counterclockwise direction. We construct a complementary graph $\overline{G_r}$ of G_r, where G_r is a undirected graph with all sensor nodes in N_{dead} connected in order of θ_i, as illustrated in Fig. 6.2. In the constructed graph $\overline{G_r}$, our goal becomes to *find a clique set with the maximum total weight, defined as the rescue sensors selection problem (RSSP).* Recall that, given a graph $G = (V, E)$, a clique set is a subset of V, any two of which are adjacent (connected by an edge). By constructing $\overline{G}_r$, we can guarantee the discovered clique set contains only non-adjacent nodes in N_{dead} (since edges connecting adjacent nodes in G_r are all removed). For the problem of finding a clique set with the maximum total weight, we recall the maximum-weight clique problem (MWCP) and formally define as follows.

Definition 1. *Given a weighted undirected graph $G = (V, E, \boldsymbol{w})$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the vertex set, $E \subseteq V \times V$ is the edge set and $\boldsymbol{w} \in \mathbb{R}^n$ is the weight vector. Each v_i has a corresponding w_i. Two distinct vertices $v_i, v_j \in V$ are adjacent if they are connected by an edge. Given a subset of vertices $V_c \subseteq V$, the weight corresponds with V_c is $\boldsymbol{w}(V_c) = \sum_{v_i \in V_c} w_i$. A clique set V_c is a subset of vertices set V, any two of which are adjacent. The MWCP is the problem that finds a clique V_c in G having maximum weight $\boldsymbol{w}(V_c)$, and the clique V_c is constructed by k_c vertices which represents the clique size.*

The MWCP is known as an NP-hard problem [18]. By defining $\overline{G}_r = (N_{dead}, \overline{E}_r, \boldsymbol{w_r})$, the MWCP can be reduced to our RSSP, proving RSSP is also NP-hard.

Theorem 2. The rescue sensors selection problem (RSSP) is NP-hard.

Proof. To prove RSSP is NP-hard, we reduce the MWCP to RSSP by showing MWCP $\leq_p$ RSSP. In other words, any instance of MWCP can be reduced in polynomial time to an instance of RSSP. Let $G = (V, E, \boldsymbol{w})$ represent an arbitrary instance of the MWCP. We can transform G to an instance of the RSSP $\overline{G}_r$ by taking $N_{dead} = V$, $\overline{E}_r = E$ and $\boldsymbol{w_r} = \boldsymbol{w}$ in polynomial time. We claim that we can find the maximum-weight clique V_c with $\boldsymbol{w}(V_c)$ for the MWCP if and only if we can find rescue sensors set R_{dead} with $\boldsymbol{w}(R_{dead})$ for the RSSP. For the **if** part, suppose that G has a maximum-weight clique $V_c \subseteq V$ with $\boldsymbol{w}(V_c) = w$ containing k_c vertices. By taking $G = \overline{G}_r$, $V = N_{dead}$, $E = \overline{E}_r$ and $\boldsymbol{w} = \boldsymbol{w_r}$, we can find a rescue sensors set $R_{dead} \subseteq N_{dead}$ with $\boldsymbol{w}(R_{dead}) = w$ containing k_c vertices. Conversely, we prove the **only if** part. Suppose that $\overline{G}_r$ has a rescue sensors set $R_{dead} \subseteq N_{dead}$ with $\boldsymbol{w}(R_{dead}) = w$ containing $|R_{dead}|$ vertices. By taking $\overline{G}_r = G$,

$N_{dead} = V$, $\overline{E}_r = E$ and $\boldsymbol{w_r} = \boldsymbol{w}$, there must exist a maximum-weight clique $V_c \subseteq V$ with $\boldsymbol{w}(V_c) = w$ containing $|R_{dead}|$ vertices, which completes the proof. $\square$

Several approximating algorithms exist to solve the MWCP in efficient computational time [6, 17]. However, only positive weights are considered in these solutions, for no efficient algorithm is available yet to handle the negative weights [20]. In light of this, and considering the high complexity of estimating w_h [19], we formulate only positive weights by using $w_i = \sum_{s_j \in N_i} w_{ij}$ in this work so that efficient algorithms can be applied to solve the RSSP. Furthermore, by adopting positive weights associated with sensor candidates in N_{dead}, we observe a nice property of RSSP that the selected rescue sensors set R_{dead} (maximum-weight clique in $\overline{G}_r$) is guaranteed to contain *exactly* $\lfloor \frac{|N_{dead}|}{2} \rfloor$ nodes (clique size of $\lfloor \frac{|N_{dead}|}{2} \rfloor$), since no negative weights are possibly to reduce the set size beyond $\lfloor \frac{|N_{dead}|}{2} \rfloor$. This is contrary to the case of arbitrary graph handled by MWCP, in which the size of maximum-weight clique is unknown (even only positive weights are considered). In $\overline{G}_r$ of our RSSP, given only positive weights, the size of maximum-weight clique (MWC) is fixed at $\lfloor \frac{|N_{dead}|}{2} \rfloor$ and the search for MWC can be easily accomplished by trying combinations of every other nodes in $\overline{G}_r$, leading to time complexity of $\Theta(|N_{dead}|)$. The obtained MWC is then selected as the rescue sensors set R_{dead}. We term this selection procedure as MWC-FS (maximum-weight clique with fixed size) approach operated on graph $\overline{G}_r$. In this way, we convert the originally intractable subject into a solvable problem, for which a suitable rescue sensors set R_{dead} can be obtained within a reasonable computation time. We intend to keep the selection mechanism at a moderate complexity for practical concerns, leaving the suboptimality caused by this imperfect selection strategy to be handled by possibly EVFA-B global redeployments.

6.2 Physical Movements Performed by Selected Rescue Sensors

Once the rescue sensors set R_{dead} is determined, we propose to perform two-tier physical movements to gradually recover the sensing hole. As displayed in Fig. 6.3, for each selected rescue sensor $s_r \in R_{dead}$, the 1^{st}-*tier* movement is applied such that the center of s_{dead} can be exactly covered by s_r. Specifically, the clusterhead instructs s_r to move toward the center of s_{dead} by the amount of Δ_r offset. Given the coordinates of s_r, s_{dead}, and sensing radius r_r, Δ_r can be easily obtained. After performing the 1^{st}-*tier* movement, some neighbors of s_r may become disconnected. For those affected immediate neighbors of s_r, we suggest to apply the 2^{nd}-*tier* movement. Suppose s_{r_i} represents some disconnected neighbor of s_r. The clusterhead then instructs s_{r_i} to move toward s_r by the amount of Δ_{r_i} offset. In this work, we attempt to restore the originally balanced distance relationship between s_r and s_{r_i}, and therefore set $\Delta_{r_i} = d(s_r, s_{r_i}) - d_{th}^{rr_i}$. All affected immediate neighbors of s_r should perform the 2^{nd}-*tier* movements, as illustrated in Fig. 6.3.

One may argue that *more-tier* physical movements with gradually decreased movement (offset) amounts should be performed. However, this complicates the computation, and does not produce significant coverage improvement in our experiments. As a result,

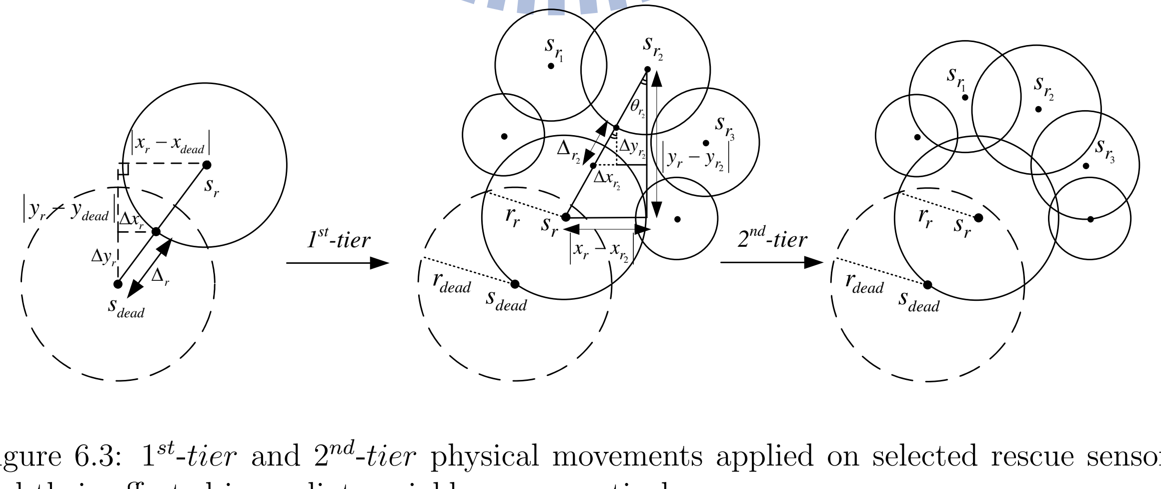


Figure 6.3: 1^{st}-*tier* and 2^{nd}-*tier* physical movements applied on selected rescue sensors and their affected immediate neighbors, respectively.

we restrict our local repairing within two tiers. For cases that are beyond the recovery capability of two-tier movements due to insufficient sensors available around the sensing hole, we simply activate EVFA-B for global redeployments.

6.3 SSOA Algorithm Summary

Experimental experiences reveal that the local recovery mechanism exercised by SSOA provides the network an effective self-healing capability in many faulty cases, where faulty sensors are generally evenly distributed across the network. In extreme cases, where sensor faults are concentrated at certain locations, leading to a reduced sensing coverage below c_{th} even after the local repairing is performed, then EVFA-B should be utilized to globally redeploy the sensors. We outline the SSOA operations by providing the pseudocode in Algorithm 3.

Algorithm 3 Sensor Self-organizing Algorithm (SSOA)

```
1: while ($s_{dead}$ detected) do
2:     evaluate $c_{now}$;
3:     if ($c_{now} < c_{th}$) then
4:         perform EVFA-B to redeploy the entire WSN;
5:     else
6:         obtain the overlapping degree $w_i$ of each $s_i \in N_{dead}$;
7:         construct graph $\overline{G}_r$;
8:         apply MWC-FS approach to determine the maximum-weight clique set in $\overline{G}_r$;
9:         rescue sensors set $R_{dead}$ is selected as the determined clique set;
10:        for each $s_r \in R_{dead}$ do
11:            perform the 1st-tier physical movement;
12:            for each affected neighbor $s_{r_i} \in N_r$ do
13:                perform the 2nd-tier physical movement;
14:            end for
15:        end for
16:    end if
17: end while
```

Chapter 7

Performance Evaluation

In this section, we validate the proposed CASA protocol by comparing the performance with two other self-deployment mechanisms in terms of coverage ratio, network self-healing capability, and total energy consumed by sensor physical movements. The comparison targets include mechanisms also based on virtual forces. We implement Zou (introduced in [31]) and Zou-B (improved Zou mechanism by incorporating boundary forces into the force calculations) with fixed weight settings. Since there is no specific design guidelines provided by [31] on setting the weights except for suggesting to use $w_r >> w_a$, we try on several w_r and w_a combinations and select $(w_a = 1, w_r = 1000)$ to be utilized by Zou and Zou-B for its best coverage performance. On the other hand, the weight settings in CASA follow the derivations presented in Chapter 4.2 and are made as $(w_a = \frac{1}{k}, w_r = k\sqrt{m^2+n^2})$. For the weight w_b associated with the boundary force (considered by both CASA and Zou-B), we use the same value set for w_r (i.e., $w_b = w_r = k\sqrt{m^2+n^2}$ in CASA and $w_b = w_r = 1000$ in Zou-B). Since there is no route planning strategy available in Zou and Zou-B, we simply assume no collisions happen and sensors can always reach their destinations accurately (though this represents a serious problem in real deployment). When faulty sensors occur, Zou and Zou-B have no local recovery technique and can only perform global redeployment on being triggered by the reduced coverage lower than c_{th},

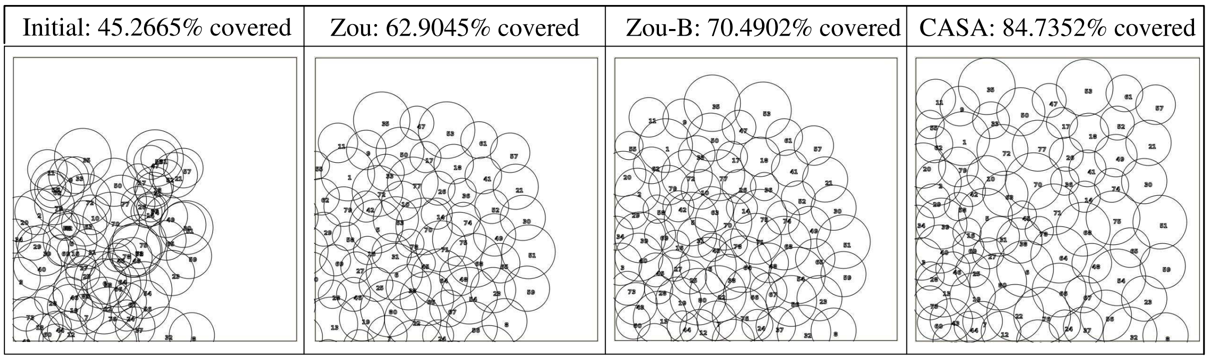


Figure 7.1: Sensor deployment status after 50 rounds (virtual movements) using Zou, Zou-B, and our proposed CASA strategies, respectively ($m = 200, n = 200, k = 80$, HSR with $\widetilde{a} \geq 1$).

while CASA is able to quickly react to the faults by constantly applying SSOA for local repairs. We simulate heterogeneous sensors, having sensing radius uniformly distributed in $[10, 20]$, in a rectangular grid-based region. The distance threshold in Zou and Zou-B is set as twice the average sensing radius (i.e., $d_{th}^{ij} = 2\overline{r}$, where $\overline{r} = \frac{1}{k}\sum_{i=1}^{k} r_i$), while CASA follows Eq. (4.3) on setting the threshold (with overlapping factor $\alpha = 0.9$). All three mechanisms use $Maxloops = 100$ and $c_{th} = 0.95$ as their deployment termination conditions.

7.1 Improved Surveillance Coverage

Fig. 7.1 displays the deployment results accomplished by Zou, Zou-B, and CASA respectively at the 50^{th} round, halfway to the maximum allowable loops of 100. We observe that, given the same computation time, CASA is able to make the most effective progress toward the required sensing coverage. On the other hand, due to lack of boundary forces, Zou makes many unnecessary movements outside the sensing field. By incorporating the boundary forces to keep sensors from drifting away, Zou-B outperforms Zou as a result of reducing unwanted coverage outside the monitoring region. However, due to improper distance threshold and weight settings, Zou-B is unable to cover the area as effectively as

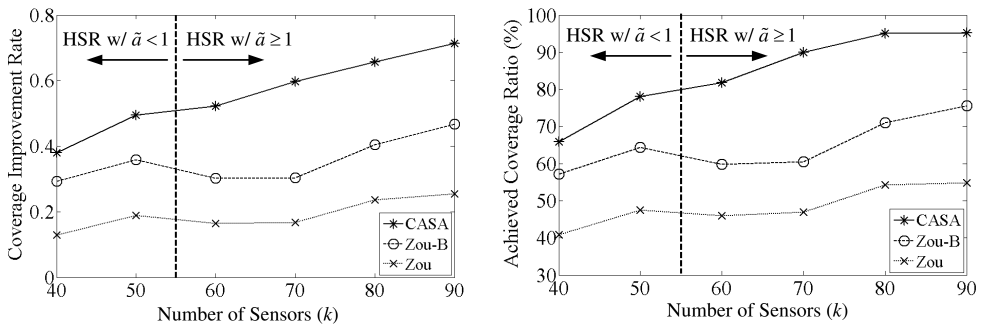


Figure 7.2: Coverage performance accomplished by Zou, Zou-B, and our CASA deployment strategies under various amounts of sensor nodes in a monitored 200×200 area. Note that the results are obtained after the first redeployment (no faulty sensor occurs yet).

CASA does.

The results in Fig. 7.1 motivate us to conduct another set of experiments investigating the *coverage improvement rate* of respective mechanism under different environmental settings. We define the coverage improvement rate as the *average amount of coverage ratio improved/increased per round/loop*, regarded as the progressing speed on enhancing sensing coverage. Since the three mechanisms have different progressing speeds, intuitively, the one with the highest coverage improvement rate is expected to produce the best coverage ratio. We experiment on various sensor populations in the same monitoring region as Fig. 7.1 to observe the coverage improvement rate and achieved coverage ratio. As shown in Fig. 7.2, after the first redeployment, CASA achieves the best sensing coverage due to its highest coverage improvement rate under all sensor populations. Moreover, we observe that both the coverage improvement rate and achieved coverage ratio of CASA increase monotonically as number of sensors grows. The reason attributes to the judicious designs of distance threshold and weight constants, making the deployment strategy adopted by CASA adaptive to environmental parameters (such as sensor numbers, area dimensions, and heterogeneous sensing ranges). On the other hand, Zou and Zou-B do not have steadily increasing performance as sensor population grows, due to their improper parameter designs, making the two mechanisms incapable of utilizing

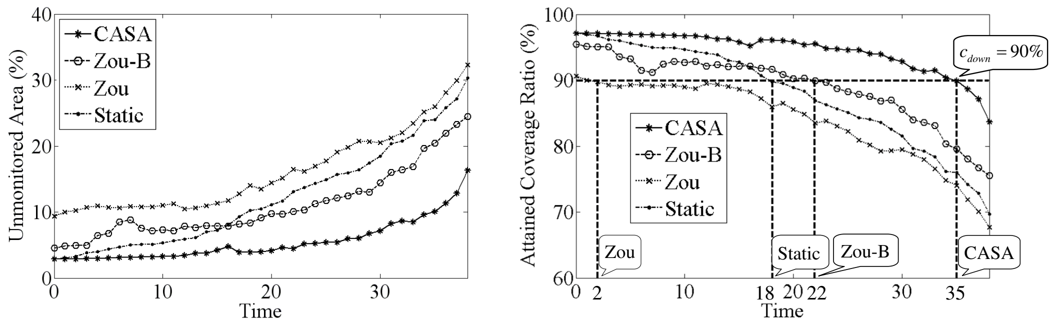


Figure 7.3: Network self-healing performance comparison in a monitored 120×120 environment with 70 sensors where some faulty sensor occurs every unit time (number of sensors reduced to only 32 at the 38^{th} time unit).

the benefit brought by increased number of deployable sensors.

7.2 Network Self-healing Capability

Once the desired sensing coverage is achieved by the first redeployment, how to maintain an effective surveillance coverage as faulty sensors occur over time is an important issue. In this section, we investigate this issue by simulating an environment where faulty sensor occurs at every time unit. We additionally implement the Static mechanism, which applies EVFA-B deployment strategy as CASA does and remains statically without any further redeployments, for comparison purpose. We observe the unmonitored area and attainable coverage ratio, as illustrated in Fig. 7.3. Due to the capability of local repairing enabled by SSOA, CASA is able to quickly react to sensor faults and recover the sensing voids. For Zou and Zou-B, the global redeployment is triggered only when the sensing coverage is reduced below c_{th}, leading to slow reactions and inefficient coverage recovery. Suppose the network is considered to be invalid/down when sensing coverage is below 90% ($c_{down} = 0.9$). Fig. 7.3 (right) depicts the operative network lifetime yielded by respective mechanism. Under the same environmental settings with the same faults occurrence behavior, CASA maintains the longest functioning time (35 time units) by its

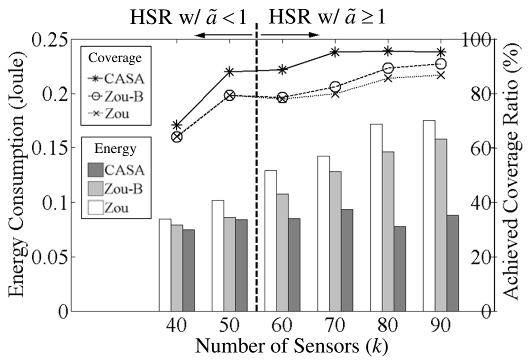


Figure 7.4: Physical movement energy consumption comparison after the first redeployment is respectively completed by Zou, Zou-B, and CASA under various amounts of sensor nodes in a monitored 200×200 area.

best network self-healing capability. We also observe that the operation time produced by Static is longer than Zou and comparable with Zou-B. This is interesting since Static only deploys the network once using our EVFA-B mechanism adopted by CASA, implying the inherently nice property of tolerating unexpected faults possessed by our proposed deployment strategy.

7.3 Energy Conservation on Physical Movements

Due to the centralized computations and communications exercised by Zou, Zou-B, and CASA, the major source of energy consumption is from sensor physical movements. To model the energy consumed by the motion device moving for one grid unit, we do real measurements on the sensor robot used in our implementation testbed with grid size equal to 1 cm. The robot assembles six 1.2 V 2000 mAh rechargeable NiMH batteries with measured $200 \sim 290$ mA moving current and average moving speed at 0.06 m/sec (216 m/hr). Consequently, the average moving energy consumption per grid (unit distance) can be estimated by $0.29 \times 7.2 \times (\frac{0.01}{216}) = 9.667 \times 10^{-5}$ Joule. We obtain the total energy consumed by physical movements performed by respective deployment strategy based on the estimated energy model, and conduct experiments to observe the energy performance under different sensor populations. Fig. 7.4 shows the results of both energy consumption and achieved coverage ratio. CASA yields the highest coverage ratio, while consuming the least energy on physical movements, due to its capability of keeping sensors from moving far away. The results indicate that CASA is both coverage effective and energy efficient, which encourages us to implement the CASA protocol suite in a practical home testbed.

Chapter 8

Implementation of an Automated Home Monitoring Network (MoNet)

As pointed out in [10] that simulation models do not sufficiently capture the radio and sensor irregularity in a real-world environment, a proof-of-concept implementation is thus needed to demonstrate the feasibility of our proposed CASA protocol. In this section, we briefly report our prototyping experiences on an automated home monitoring network

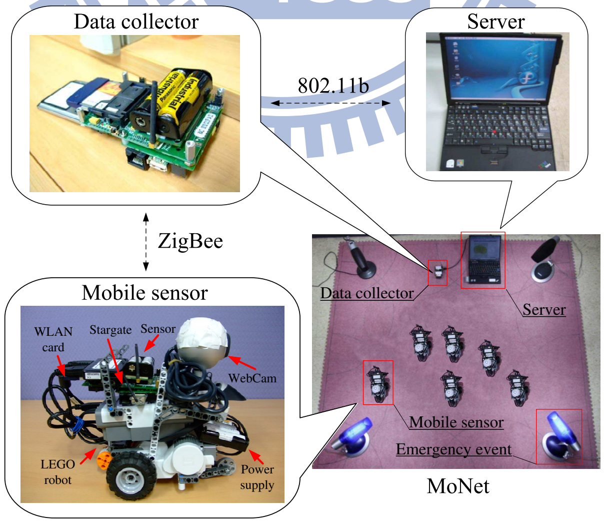


Figure 8.1: Validation of the proposed CASA protocol suite by implementing a real-world home monitoring network (MoNet) via commodity hardware components.

(MoNet) enabled by CASA.

Fig. 8.1 illustrates the hardware architecture and communication protocols used by our MoNet. the mobile sensor is basically a moving robot (LEGO MINDSTORMS NXT 9797 [2]) carrying a single-board computer (Crossbow Stargate [1]), a sensor-equipped mote (Crossbow MICAz [1]), and a webcam device (Logitech QuickCam Pro 4000 [3]). The server acts as the clusterhead performing deployment-related computations required by CASA, while the data collector is responsible for gathering necessary data (such as sensor locations and sensing ranges) from all sensors via ZigBee protocol and providing them to the server. In our testbed, the location information is obtained via a pre-deployed RFID positioning system with grid granularity of 1 cm. To demonstrate the emergency response capability of MoNet, we randomly place six mobile sensors in a $2m \times 2m$ area, and generate four emergency events (using desk lamps instead of real fire for safety concerns) at the four corners, as shown in Fig. 8.2. We configure the sensors to regard a light event with reading above 900 as an abnormal event (emergency) and report the detected event back to the server upon the detection. In addition, we simulate faulty sensors by turning off s_1 and s_2 at demonstration time snapshots t_1 and t_2 respectively, leading to more detection holes as time advances, to test the network self-healing competency. As

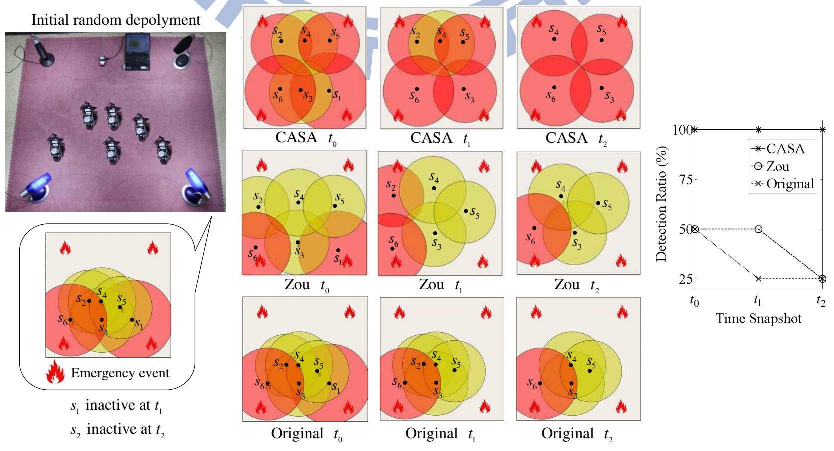


Figure 8.2: Performance results obtained from our home MoNet prototype.

revealed in Fig. 8.2, the Original mechanism represents that none of redeployment or self-healing strategies is applied to improve the detection ratio, while CASA is always able to detect all the four emergency events even in the face of faulty sensors. The results obtained from our MoNet testbed further justify the CASA designs. A brief demonstration video on this experiment is available in [4].

Chapter 9

Conclusions

In this thesis, we propose a coverage-aware sensor automation (CASA) protocol with the objective of providing effective surveillance coverage for the home environment. Three centralized algorithms are included in the CASA protocol suite, namely EVFA-B, CFPP, and SSOA, to separately handle the global sensor self-deployment, sensor moving path scheduling when executing self-deployment, and sensor self-organization in the presence of node failures. By the current definition of sensing coverage, we adopt the 1-covered detection model (area considered fully monitored if every grid point is covered by at least one sensor). To enhance the surveillance reliability, one may expect to have k-covered sensing model (every grid point is covered by at least k sensors). Such extension can be generally achieved by decreasing the distance threshold values to allow a certain level of sensing redundancy (though specific relationship between the threshold value adjustment and attainable coverage degree still need be further characterized). In this work, we attempt to realize a practical home surveillance system by addressing the sensor deployment-related problems in a unified framework. An automated home monitoring network (MoNet) powered by our proposed CASA protocol set is implemented as a proof-of-concept prototype to corroborate the protocol feasibility and demonstrate the emergency detection capability of MoNet.

Bibliography

[1] Crossbow Technology. `http://www.xbow.com/`.

[2] LEGO MINDSTORMS NXT 9797. `http://www.lego.com/en-US/default.aspx`.

[3] Logitech QuickCam Pro 4000. `http://www.logitech.com/`.

[4] MoNet Demonstration Video. `http://bunlab.twbbs.org/filezone/files/CASA.mpg`.

[5] E. S. Biagioni and K. W. Bridges. "The Application of Remote Sensor Technology to Assist the Recovery of Rare and Endangered Species". *Int'l Journal of High Performance Computing Applications*, 16(3):315–324, 2002.

[6] I. M. Bomze, M. Pelillo, and V. Stix. "Approximating the Maximum Weight Clique Using Replicator Dynamics". *IEEE Transactions on Neutral Networks*, 11(6):1228–1241, November 2000.

[7] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho. "Grid Coverage for Surveillance and Target Location in Distributed Sensor Networks". *IEEE Transactions on Computers*, 51(12):1448–1453, December 2002.

[8] S. S. Dhillon and K. Chakrabarty. "Sensor Placement for Effective Coverage and Surveillance in Distributed Sensor Networks". *In Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1609–1614, March 2003.

[9] A. Howard, M. J. Mataric, and G. S. Sukhatme. "Mobile Sensor Network Deployment Using Potential Fields: A Distributed Scalable Solution to the Area Coverage Problem". *In Proc. Int'l Symposium on Distributed Autonomous Robotics Systems*, June 2002.

[10] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau. "Mobile Emulab: A Robotic Wireless and Sensor Network Testbed". *In Proc. IEEE INFOCOM*, April 2006.

[11] E. Jovanov, D. Raskovic, J. Price, J. Chapman, A. Moore, and A. Krishnamurthy. "Patient Monitoring Using Personal Area Networks of Wireless Intelligent Sensors". *Biomedical Sciences Instrumentation*, 37:373–378, 2001.

[12] F. Y. S. Lin and P. L. Chiu. "A Near-optimal Sensor Placement Algorithm to Achieve Complete Coverage/Discrimination in Sensor Networks". *IEEE Communications Letters*, 9(1):43–45, January 2005.

[13] M. Locatelli and U. Raber. "Packing Equal Circles in a Square: A Deterministic Global Optimization Approach". *Elsevier Discrete Applied Mathematics*, 122(1-3):139–166, October 2002.

[14] J. Lu and T. Suda. "Differentiated Surveillance for Static and Random Mobile Sensor Networks". *IEEE Transactions on Wireless Communications*, 7(11):4411–4423, November 2008.

[15] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. "Wireless Sensor Networks for Habitat Monitoring". *In Proc. Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002.

[16] S. A. Musman, P. E. Lehner, and C. Elsaesser. "Sensor Planning for Elusive Targets". *Elsevier Mathematical and Computer Modelling*, 25(3):103–115, February 1997.

[17] P. R. Ostergard. "A New Algorithm for the Maximum-weight Clique Problem". *Nordic Journal of Computing*, 8(4):424–436, December 2001.

[18] P. M. Pardalos and J. Xue. "The Maximum Clique Problem". *Journal of Global Optimization*, 4(3):301–328, April 1994.

[19] K.-H. Peng, T.-Y. Lin, and K.-H. Chen. "On Seamless Wireless Sensor Deployment and System Implementation". *Int'l Journal of Advanced Information Technologies (IJAIT)*, 3(2):72–92, December 2009.

[20] B. Rosgen and L. Stewart. "Complexity Results on Graphs with Few Cliques". *Discrete Mathematics and Theoretical Computer Science*, 9(1):127–136, July 2007.

[21] O. Tekdas, V. Isler, J. H. Lim, and A. Terzis. "Using Mobile Robots to Harvest Data from Sensor Fields". *IEEE Wireless Communications*, February 2009.

[22] D. Tian and N. D. Georganas. "A Coverage-preserving Node Scheduling Scheme for Large Wireless Sensor Networks". *In Proc. ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 32–41, September 2002.

[23] Y.-C. Tseng, Y.-C. Wang, and K. Cheng. "An Integrated Mobile Surveillance and Wireless Sensor (iMouse) System and Its Detection Delay Analysis". *In Proc. ACM Int'l Conf. on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWiM)*, 2005.

[24] G. Wang, G. Cao, and Thomas F. La Porta. "Movement-assisted Sensor Deployment". *IEEE Transactions on Mobile Computing*, 5(6):640–652, June 2006.

[25] Q. Wang, K. Xu, G. Takahara, and H. Hassanein. "Device Placement for Heterogeneous Wireless Sensor Networks: Minimum Cost with Lifetime Constraints". *IEEE Transactions on Wireless Communications*, 6(7):2444–2453, 2007.

[26] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. ”Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks”. *In Proc. ACM Sensys*, November 2003.

[27] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng. ”Efficient Placement and Dispatch of Sensors in a Wireless Sensor Network”. *IEEE Transactions on Mobile Computing*, 7(2):262–274, February 2008.

[28] J. Wu and S. Yang. ”SMART: A Scan-based Movement-assisted Sensor Deployment Method in Wireless Sensor Networks”. *In Proc. IEEE INFOCOM*, March 2005.

[29] F. Ye, G. Zhong, J. Cheng, S. Lu, and L. Zhang. ”PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks”. *In Proc. IEEE Int’l Conf. on Distributed Computing Systems (ICDCS)*, pages 28–37, May 2003.

[30] H. Zhang and J. C. Hou. ”Maintaining Sensing Coverage and Connectivity in Large Sensor Networks”. *Ad Hoc and Sensor Wireless Networks*, 1(1-2):89–124, March 2005.

[31] Y. Zou and K. Chakrabarty. ”Sensor Deployment and Target Localization Based on Virtual Forces”. *In Proc. IEEE INFOCOM*, April 2003.