國立交通大學

資訊工程系

博 士 論 文

在無線感測網路中的覆蓋及連結問題

The Coverage and Connectivity Problems in Wireless Sensor Networks

研 究 生：黃啟富

指導教授：曾煜棋　教授

中 華 民 國 九 十 三 年 十 月

在無線感測網路中的覆蓋及連結問題
The Coverage and Connectivity Problems in Wireless Sensor Networks

研 究 生：黃啟富　　Student：Chi-Fu Huang

指導教授：曾煜棋　　Advisor：Yu-Chee Tseng

國 立 交 通 大 學
資 訊 工 程 系
博 士 論 文

A Dissertation
Submitted to Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in

Computer Science and Information Engineering

October 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年十月

在 無 線 感 測 網 路 中 的 覆 蓋 及 連 結 問 題

學生：黃啟富　　　　　　　　　　　　　　　　　指導教授：曾煜棋

國立交通大學資訊工學系博士班

摘　　要

無線感測網路的新興技術，將使得我們的生活更加便利，它的環境之中包含了許多便宜的無線感測器，每一個感測器都具有收集、儲存、以及處理環境資訊的能力，此外，感測器和感測器之間還能透過無線的方式互相通訊。要使得一個感測網路可以成功的運行，感測器必須要能夠同時保持感測的覆蓋程度、網路的連結性、以及長時間運行的能力。在本論文之中，我們首先探討在二維及三維空間之中的覆蓋問題，接著，我們研究在一個感測網路中，感測的覆蓋程度和通訊的連結性之間的關係，然後，為了延長網路的運行時間，我們提出了分散式的能源節省暨覆蓋維持協定。

覆蓋問題是無線感測網路的基礎問題之一，它反應了一個無線感測網路是如何地被感測器所監看或追蹤。在本論文之中，我們把這個問題定義成一個判定問題：給定一個參數 k，我們的目標是要去判定，是不是在一個感測網路所服務的區域上的每一位置，都能被最少 k 個感測器所偵測到。在二維的平面上，每一個感測器的感測範圍可以是相同或不同半徑的圓形，而在三維的立體空間之中，我們將感測器的感測區域假設成圓球(不必是相同的半徑)，我們首先提出這個問題在二維空間之中多項式時間的演算法，接著，我們更進一步的証明這個問題在三維空間之中，也可以在多項式的時間之內解答，而我們所提出來的解法，都可以很容易地轉換成分散式的協定。

要讓一個感測網路可以順利的運行，感測器必須要能夠同時兼顧感測的覆蓋程度以及網路的連結性，這個問題己經在[35, 43]之中被探討，這兩篇

文獻都提出了一個類似的結論，那就是，當通訊的半徑大於等於兩倍的感測半徑時，覆蓋的程度可保証網路的連結性。在本論文中，我們從另外一個角度來探討這個問題，並提出了不依賴上面假設的解決方案，這也暗示著，[35, 43]所提出的方法，可以被歸類成我們所提出的方法之中的一個特別情況。

此外，保持足夠的覆蓋程度和達成系統長時間的存活，是在設計網路的拓樸時兩個互相衝突的因素，在本論文中，我們提出分散式的協定來安排感測器工作及休眠時間，可以有效率的提升網路的存活時間，並同時維持被偵測區域足夠的覆蓋，所提出的協定類似[40]所提的模型，然而，我們的方法可以大量的減少計算複雜度，以及平衡感測器之間電能的消耗。

The Coverage and Connectivity Problems in Wireless Sensor Networks

student：Chi-Fu Huang Advisors：Dr. Yu-Chee Tseng

Department of Computer Science and Information Engineering
National Chiao Tung University

ABSTRACT

The wireless sensor network is an emerging technology that may greatly facilitate our life. Such environments may have many inexpensive wireless nodes, each capable of collecting, storing, and processing environmental information, and communicating with neighboring nodes. For a sensor network to operate successfully, sensors must maintain sensing coverage, network connectivity, and long lifetime. In this dissertation, we first study the coverage problems both in 2D and 3D spaces. Next, we investigate the relationship between sensing coverage and communication connectivity of a sensor network. Then, decentralized energy-conserving and coverage-preserving protocols are proposed to prolong the network lifetime.

One of the fundamental issues in sensor networks is the *coverage* problem, which reflects how well a sensor network is monitored or tracked by sensors. In this dissertation, we formulate this problem as a decision problem, whose goal is to determine whether every point in the service area of the sensor network is covered by at least *k* sensors, where *k* is a given parameter. The sensing ranges of sensors in a 2D space can be unit disks or non-unit disks while in a 3D space the sensing regions of sensors are modeled by balls (not necessarily of the same radius). We first present polynomial-time algorithms, in terms of the number of sensors, to solve this problem in a 2D space. Next, we show that tackling this problem in a 3D space is still feasible within polynomial time. The proposed solutions can be easily translated into efficient polynomial-time distributed protocols.

For a sensor network to operate successfully, sensors must maintain both sensing coverage and network connectivity. This issue has been studied in [35, 43], both of which reach a similar conclusion that coverage can imply connectivity as long as sensors' communication ranges are no less than twice their sensing ranges. In this dissertation, we investigate this issue from a different angle and propose more general decentralized solutions that do not rely on the above assumption. Hence, the results in [35, 43] can be regarded as special cases of what proposed in this dissertation.

Besides, to maintain sufficient coverage and to achieve long system lifetime are two contradicting factors in designing the topology of a network. In this dissertation, we propose decentralized protocols that schedule sensors' active and sleeping periods to prolong the network lifetime while maintain the sensing field sufficiently covered. The proposed protocols are similar to the model in [40]. However, our approach can significantly reduce the computational complexity incurred, and balance the energy expenditure among sensors.

誌　　謝

感謝我所擁有的一切

感謝我的指導教授 曾煜棋 教授授辛勤的指導，感謝家人的支持，感謝實驗室中的每一個成員，感謝所有和我共事過的學長姐和學弟妹。

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks* possible. Such environments may have many inexpensive wireless nodes, each capable of collecting, storing, and processing environmental information, and communicating with neighboring nodes. In the past, sensors are connected by wire lines. Today, this environment is combined with the novel *ad hoc* networking technology to facilitate inter-sensor communication [26, 31]. The flexibility of installing and configuring a sensor network is thus greatly improved. Recently, a lot of research activities have been dedicated to sensor networks, including design issues related to the physical and media access layers [29, 38, 42] and routing and transport protocols [4, 8, 12]. Localization and positioning applications of wireless sensor networks are discussed in [3, 5, 22, 27, 33].

Since sensors may be spread in an arbitrary manner, a fundamental issue in a wireless sensor network is to ensure *coverage* and *connectivity*. Given a sensor network, the coverage problem in general is to determine how well the sensing field is monitored or tracked by sensors. In the literature, this problem has been formulated in various ways. Even in computational geometry, some coverage-related solutions can be found [24, 37]. Although solutions to those problems cannot be directly applied to wireless sensor networks, it is still worth to study those problems to establish some theoretical backgrounds on the coverage problem. Indeed, a lot of works have been dedicated to the coverage-related problems in wireless sensor networks in the last few years. These include the surveillance and exposure of sensor networks [19, 20, 34], and the concerns of coverage versus connectivity issues when deploying a sensor network [9, 28, 35, 43].

The connectivity issue is concerned with the diversity of communication paths between sensors. This would affect network robustness and communication performance. The *GAF* protocol [39] aims to extend the network lifetime by turning off redundant nodes while keeping the same level of *routing fidelity*, which is defined as uninterrupted connectivity between communicating nodes. GAF imposes a virtual grid on the network and nodes in the same grid coordinate with each other to determine who can sleep and how long. Reference [7] presents a connectivity-maintaining protocol, *SPAN*, which can turn off unnecessary nodes such that all active nodes are connected through a communication backbone and all inactive nodes are directly connected to at least one active node. Maintaining a network connected is also a basic requirement of works targeted at *topology control*, which is to adjust sensors' transmission power for energy efficiency and collision avoidance [6, 15, 36].

Another concern in a wireless sensor network is the energy issue. Since sensors are usually powered by batteries, sensors' on-duty time should be properly scheduled to conserve energy. If some nodes share the common sensing region and task, we can turn off some of them to conserve energy and thus extend the lifetime of the network. This issue has been extensively studied recently. Reference [30] proposes a heuristic to select mutually exclusive sets of sensor nodes such that each set of sensors can provide a complete coverage of the monitored area. Also targeted at turning off some redundant nodes, [41] proposes a probe-based density control algorithm to put some nodes in a sensor-dense area to a doze mode to ensure a long-lived, robust sensing coverage. A coverage-preserving node scheduling scheme based on an eligibility rule which allows a node to turn itself off as long as other neighboring nodes can cover its sensing area is presented in [32]. A coverage-aware self-scheduling scheme based on a probabilistic sensing model is proposed in [17]. The work [40], which tries to fairly distribute energy consumption among sensors, is proposed. The whole sensing area is divided into grid points which are used to evaluate whether the area is sufficiently covered or not. Each sensor has to join the schedule of each grid point covered by itself such that the grid point is covered by at least one sensor at any moment.

In this dissertation, we first study the coverage problems both in 2D and 3D spaces. Furthermore, the relationship between sensing coverage and communication connectivity of a sensor network is investigated. Finally, to prolong the network lifetime, decentralized coverage-preserving node-scheduling protocols are proposed.

We consider a more general sensor coverage problem in this dissertation: Given a set of sensors deployed in a target area, we want to determine if the area is sufficiently *k-covered*, in the sense that every point in the target area is covered by at least k sensors, where k is a given parameter. As a result, the aforementioned works [32, 41] can be regarded as a special case of this problem with $k = 1$. Applications requiring $k > 1$ may occur in situations where a stronger environmental monitoring capability is desired, such as military applications. It also happens when multiple sensors are required to detect an event. For example, the triangulation-based positioning protocols [22, 27, 33] require at least three sensors (i.e., $k \geq 3$) at any moment to monitor a moving object. Enforcing $k \geq 2$ is also desirable for fault-tolerant purpose. The *arrangement* issue [2, 11], which is widely studied in combinatorial and computational geometry, also considers how a finite collection of geometric objects decomposes a space into connected elements. However, to construct arrangements, only *centralized* algorithms are proposed in the literature, whilst what we need for a wireless sensor network is a distributed solution. The solutions proposed in this dissertation can be easily translated to a distributed protocol where each sensor only needs to collect local information to make its decision.

In a 2D space, the sensing range of each sensor can be a unit disk or a non-unit disk. Rather than determining the coverage of each location, our approach tries to look at how the perimeter of each sensor's sensing range is covered, thus leading to an efficient polynomial-time algorithm. As long as the perimeters of sensors are sufficiently covered, the whole area is sufficiently covered.

In a 3D space, the sensing region of each sensor is modeled by a 3D ball. At the first glance, the 3-dimensional coverage problem seems very difficult since even determining the subspaces divided by the spheres of sensors' sensing ranges is very complicated. However, in this dissertation, we show that tackling this problem is still feasible within polynomial time. We propose a novel solution by reducing the geometric problem from a 3D space to a 2D space, and further to a 1D space, thus leading to a very efficient solution. In essence, our solution tries to look at how the sphere of each sensor's sensing range is covered. As long as the spheres of all sensors are sufficiently covered, the whole sensing field is sufficiently covered. To determined whether each sensor's sphere is sufficiently covered, we in turn look at how each spherical cap and how each circle of the intersection of two spheres is covered. By stretching each circle on a 1-dimensional line, the level of coverage can be easily determined.

In this dissertation, we further study the relationship between sensing coverage and communication connectivity of a sensor network. Reference [35] claims that coverage can imply connectivity as long as sensors' communication ranges are no less than twice their sensing ranges. A similar result is also drawn in [43]. It is clear that the results in [35, 43] are not applicable when some sensors' communication ranges are less than twice their sensing ranges even though others are not. Also, both [35, 43] assume that all sensors have the same sensing ranges. In this dissertation, we relax these constrains and show conditions for a sensor network to be k-covered and k-connected, and to be k-covered and 1-connected. Hence, the results in [35, 43] can be regarded as special cases of what proposed in this dissertation.

Finally, we propose decentralized energy-conserving and coverage-preserving protocols to prolong the network lifetime. We adopt the model in [40] to fairly distribute sensors' energy expenditure. However, instead of using grid points, we utilize the result in [35] by calculating sensors' schedules based on the intersection points among their sensing ranges. The result can significantly reduce the computational complexity incurred on each sensor. In addition, the inaccuracy problem caused by gird approximation is completely eliminated. Besides, we further discuss how to utilize sensors' remaining energy to adjust parameters in our protocol to balance the energy consumption among sensors. Simulations are conducted to verify our results.

This dissertation is organized as follows. Related works are surveyed in Chapter 2. Chapter 3 formally defines the coverage problems in a two-dimensional space and then presents our solutions while the three dimensional coverage problem is discussed in Chapter 4. Chapter 5 further studies how to ensure both coverage and connectivity. Coverage-preserving and node-scheduling protocols are presented in Chapter 6. Chapter 7 draws our conclusions and future works.

Chapter 2

Related Works

In this chapter, we first study several relevant computational geometric problems. Then, a number of papers aimed at solving the coverage problem in wireless sensor networks are discussed. We will address issues such as surveillance and exposure of sensor networks, coverage and connectivity in network deployment, and coverage- and energy-preserving protocols for sensor networks.

2.1 Related Geometric Problems

In this section, we review two computational geometric problems which are related to the coverage problem in a sensor network. The first problem is the *Art Gallery Problem* [24]. Imagine that the owner of an art gallery wants to place cameras in the gallery such that the whole gallery is thief-proof. There are two questions to be answered in this problem: (i) how many cameras

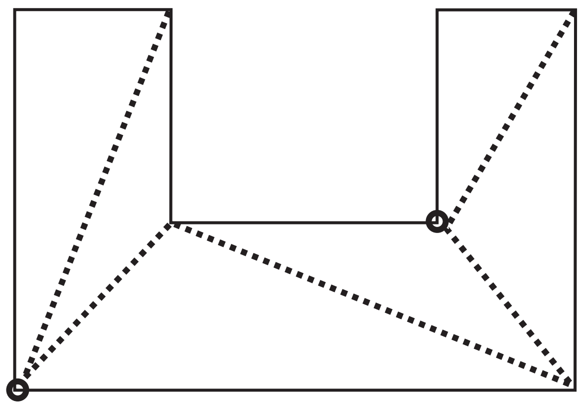


Figure 2.1: An example of triangulating a polygon and a possible deployment of cameras. Circles represent positions of cameras.

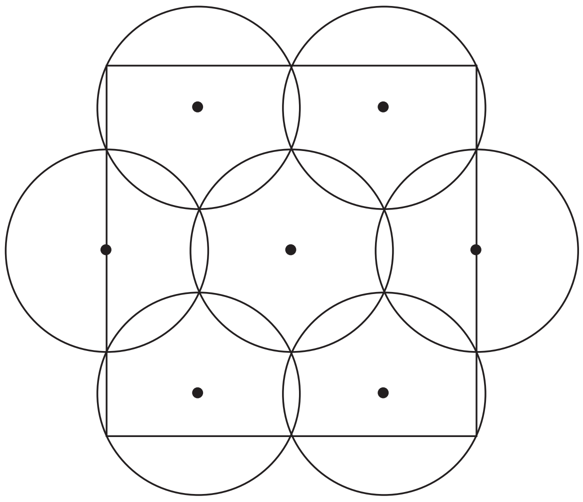


Figure 2.2: An example of an optimal covering with 7 circles. The radius of each circle is about 0.2742918.

are needed, and (ii) where these cameras should be deployed. Every point in the gallery should be monitored by at least one camera. Cameras are assumed to have a viewpoint of 360 degrees and rotate at an infinite speed. Moreover, a camera can monitor any location as far as nothing is in the middle, i.e., a line-of-sight exists. The number of cameras used should be minimized. The gallery is usually modeled as a simple polygon on a 2D plane. A simple solution to this problem is to divide the polygon into non-overlapping triangles and place one camera in each of these triangles. By *triangulating* the polygon, it has been shown that any simple polygon can be guarded by $\lfloor n/3 \rfloor$ cameras, where n is the number of triangles in the polygon. This is also the best result in the worst case [24]. An example of triangulating a simple polygon is shown in Fig. 2.1 and two cameras are sufficient to cover the gallery. Although this problem can be solved optimally in a 2D plane, it is shown to be NP-hard when being extended to a 3D space [25].

Another related problem in computational geometry is the *circle covering problem* [37], which is to arrange identical circles on a plane that can fully cover the plane. Given a fixed number of circles, the goal is to minimize the radius of circles. This issue is discussed in [13, 21, 23] for the covering of a rectangle. The coverings with less than or equal to five circles and seven circles can be done optimally [13]. For example, an optimal covering of seven circles is shown in Fig. 2.2. Reference [21] shows the coverings of six and eight circles and presents a new covering with eleven circles by an approach based on the simulated annealing. Table 2.1 lists the minimun radius r_n to cover a unit square with n identical circles reported in [23] for

Table 2.1: The minimum radius r_n to cover a unit square by n circles.

n	r_n	n	r_n	n	r_n	n	r_n
1	0.7071067...	2	0.5590169...	3	0.5038911...	4	0.3535533...
5	0.3261605...	6	0.2987270...	7	0.2742918...	8	0.2603001...
9	0.2306369...	10	0.2182335...	11	0.2125160...	12	0.2022758...
13	0.1943123...	14	0.1855105...	15	0.1796617...	16	0.1694270...
17	0.1656809...	18	0.1606396...	19	0.1578419...	20	0.1522468...
21	0.1489537...	22	0.1436931...	23	0.1412448...	24	0.1383028...
25	0.1335487...	26	0.1317468...	27	0.1286335...	28	0.1273175...
29	0.1255535...	30	0.1220368...				

$n = 1 \dots 30$.

The above geometrical computation problems are similar to the nature of coverage problems in wireless sensor networks: we need to know whether an area is sufficiently covered and monitored. The number of sensors is important in terms of cost. These results also provide some theoretical backgrounds to the coverage issue. However, there are several reasons which make solutions of geometric problems not directly applicable to wireless sensor networks. The first reason is that the assumptions are different. For example, a camera in the Art Gallery Problem can see infinite distance unless there is an obstacle. On the contrary, sensors in fact have their maximal sensing ranges. Besides, a sensor network usually has no fixed infrastructure and its topology may even change at any time. Thus, many decisions have to be made in a distributed manner. However, most geometric problems are solved in a centralized manner.

2.2 The Breach and Support Paths

In a sensor network, coverage can be regarded as the path between a given pair of points in the sensing field that is best or worst monitored by sensors when an object traverses along the path. Reference [18] defines the *maximal breach path* and the *maximal support path* as paths on which the distance from any point to the closest sensor is maximized and minimized, respectively. Polynomial-time algorithms are proposed to find such paths. The key idea is to use the Voronoi diagram and the Delaunay triangulation of sensor nodes to limit the search space. The Voronoi diagram is formed from the perpendicular bisectors of lines that connect

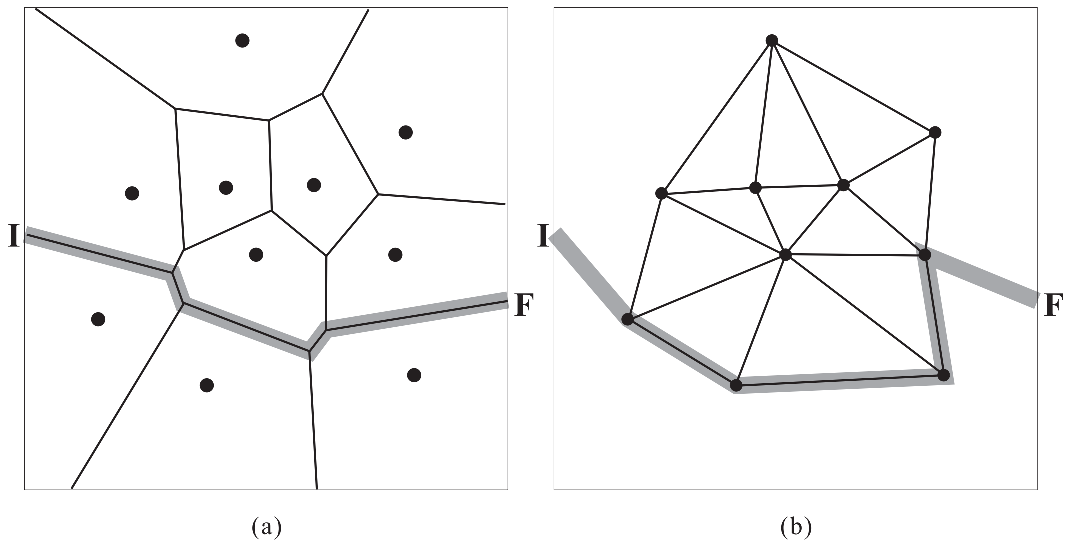


Figure 2.3: Examples of (a) the Voronoi diagram and the maximal breach path, and (b) the Delaunay triangulation and the maximal support path. I and F are the source and destination points

two neighboring sensors, while the Delaunay triangulation is formed by connecting nodes that share a common edge in the Voronoi diagram. Examples of the Voronoi diagram and Delaunay triangulation are shown in Fig. 2.3.

Because line segments in the Voronoi diagram have the maximal distance to the closest sensors, the maximal breach path must lie on the line segments of the Voronoi diagram. To find the maximal breach path, each line segment is given a weight equal to its minimum distance to the closest sensor. The proposed algorithm then performs a binary search between the smallest and largest weights. In each step, a breadth-first-search is used to check the existence of a path from the source point to the destination point using only line segments with weights that are larger than the search criterion. If a path exists, the criterion is increased to further restrict the lines considered in the next search iteration. Otherwise, the criterion is decreased. An example of the maximal breach path is shown in Fig. 2.3(a). Similarly, since the Delaunay triangulation produces triangles which have minimal edge lengths among all possible triangulations, the maximal support path must lie on the lines of the Delaunay triangulation. To find the maximal support path, line segments are assigned weights equal to their lengths . The search part is then similar to the above case. An example is shown in Fig. 2.3(b).

Reference [16] proposes decentralized algorithms to find the maximal support path by constructing the Delaunay triangulation locally. The authors claim that the Delaunay triangulation

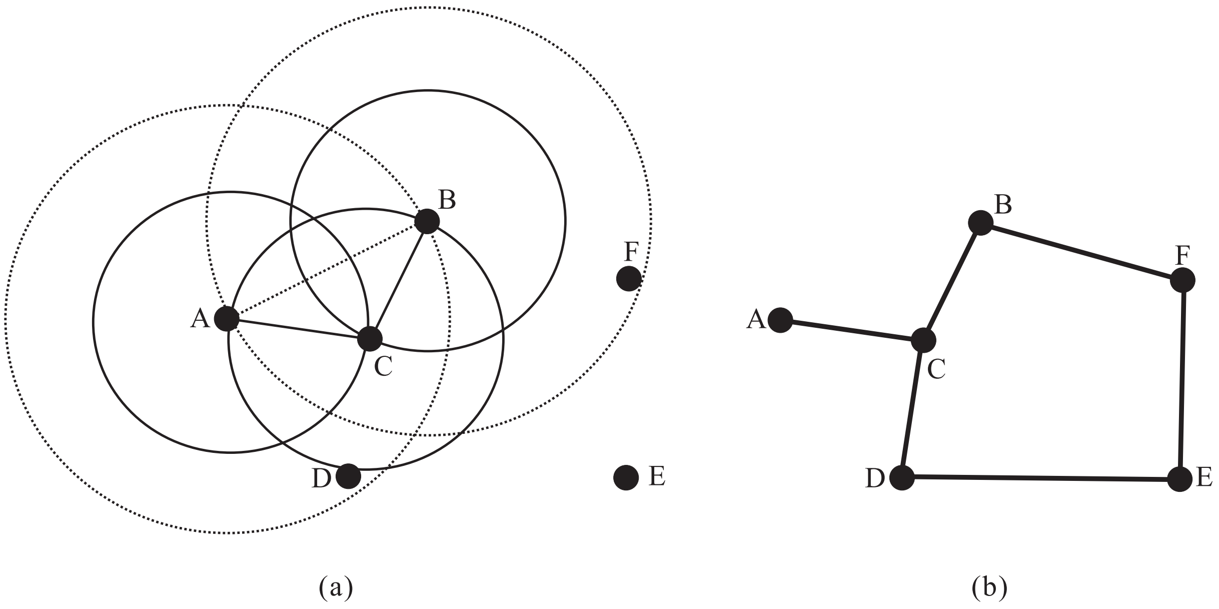


Figure 2.4: A RNG example.

can be replaced by *relative neighborhood graph (RNG)* and the maximal support path can still be found. The key idea is that the RNG can be constructed locally. Given any two sensors u and v, $lune(u, v)$ is the intersection of the two disks centered at u and v, both of the same radius $\|u, v\|$, where $\|u, v\|$ is the distance between u and v. If $lune(u, v)$ does not contain any sensor, an edge is established to join u and v with a weight of $\frac{1}{2}\|u, v\|$. The RNG is constructed by such weighted edges and all sensors. Note that RNG can be constructed distributedly by all sensors.

Fig. 2.4 shows an example of constructing a RNG. There are six sensors A, B, C, D, E, and F. In Fig. 2.4(a), $lune(A, B)$ contains node C, so the link $\overline{AB}$ should not be built. On the contrary, $lune(A, C)$ and $lune(B, C)$ do not contain any other nodes, so the links $\overline{AC}$ and $\overline{BC}$ should be built. The final RNG is shown in Fig. 2.4(b).

After constructing the RNG, a decentralized algorithm is used to find the maximal support path. First, the starting and ending points have to connect to the sensors that are closest to them. Then, the Bellman-Ford algorithm is adopted to find the minimum weight path connecting the starting and ending points. It is possible that there are several maximal support paths. In this case, the path that consumes the least energy is selected.

It is proved in [16] that the maximal support path that can be found using the Gabriel graph can also be found using the RNG. Using both Gabriel and RNG has the same complexity, $O(n \log n)$, where n is the number of sensor nodes. Thus, this conquers the problem with

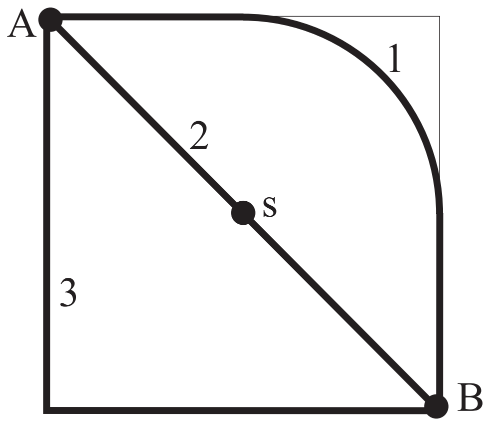


Figure 2.5: An example of exposure.

Delaunay triangulation that global information needs to be collected.

2.3 Exposure to Sensors

The concept of time should also be included to reflect how much a moving target is exposed to sensors. The exposure time should be accounted for. Consider the example in Fig. 2.5. Suppose that s is a sensor and an object moves from point A to point B at a constant speed via three possible paths. Although path 3 is the farthest path from s, it is also the longest exposure time. In contract, path 2 is the shortest path, but it has the strongest sensing intensity. Path 1 has neither the longest exposure time, nor the strongest sensing intensity.

How to find the *minimal exposure path* is addressed in [19]. The exposure for an object in the sensor field during an interval $[t_1, t_2]$ along a path $p(t)$ is defined as:

$$E(p(t), t_1, t_2) = \int_{t_1}^{t_2} I(F, p(t)) \left| \frac{dp(t)}{dt} \right| dt,$$

where $I(F, p(t))$ is the sensor field intensity measured at location $p(t)$ from the closest sensor or all sensors in the sensor field F, and $\left|\frac{dp(t)}{dt}\right|$ is the arc length. A numerical approximation is proposed in [19] to find the minimal exposure path by dividing the sensor network region into grids and forcing the path to only pass the edges of girds and/or the diagonals of grids. Each line segment is assigned a weight equal to the exposure of this segment. Then a single-source-shortest-path algorithm is used to find the minimal exposure path.

Reference [20] further discusses how to compute the exposure of a sensor network in a distributed manner. The key idea is to use the Voronoi diagram to partition the sensor field and then each sensor is responsible for the calculation of exposure in its region. Inside each region,

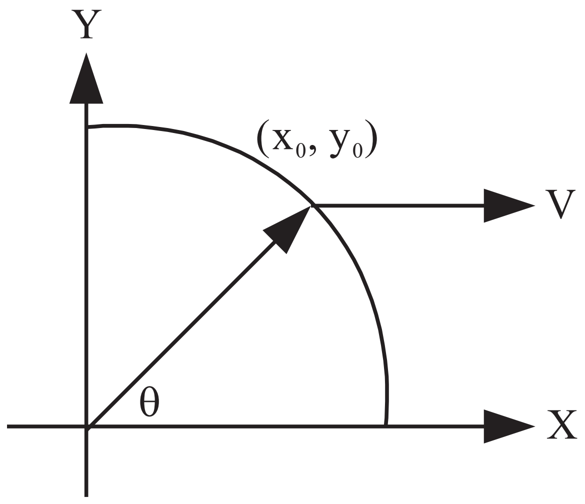


Figure 2.6: Moving direction of an object with respect to a sensor at the origin.

the above gird approximation is used. Reference [14] proposes another method to find the minimal exposure path using *variational calculus*. This work first studies the sensing field with a single sensor and then discusses how to extend the result to a more general case. Reference [34] further proposes a localized algorithm which can reduce the computational complexity of [20]. It is also proved in [34] that finding the maximal exposure path is NP-hard. Several heuristics are then provided to find the maximal exposure path.

Another important question in sensor networks is to estimate the number of sensors required to achieve complete coverage of a desired area. Reference [1] defines the *critical density thresholds* for complete coverage. The exposure is used to find the critical number of sensor nodes required to cover an area. For a sensor s, the signal received from a target decreases as the distance from the target increases. If the signal strength is less than the noise signal, the sensor can not detect the target. The authors then investigate the influence of sensors and define two radii: *radius of complete influence* (r_1) and *radius of no influence* (r_2). Objects within the former radius are surely detected and objects outside the latter radius are undetectable. If $r_1 = r_2$, the decision degenerates to the zero-one model.

Suppose that there is a sensor located at the origin. Without loss of generality, [1] assumes that the target is initially located at (x_0, y_0) on the arc at an angle θ from the x-axis, as shown in Fig. 2.6, and moves in a straight line with a constant speed v parallel to the x-axis. At a period of T, the exposure value is

$$E_s = \int_0^T \frac{\lambda}{(x_0 + v * t)^2 + (y_0)^2} dt,$$

where λ is a constant value depending on the sensor property. The exposure value can be written using polar coordinates as follows:

$$E_s = \frac{\lambda}{v\sin(\theta)} \tan^{-1}\left(\frac{\delta\sin(\theta)}{r+\delta\cos(\theta)}\right),$$

where δ is the travelling distance. Let E_{th} be the object detection threshold. It is shown in [1] that r_1 can be given bye the following equation

$$E_{th} = \frac{\lambda}{vr_1}\left(\frac{\delta}{\delta+r_1}\right)$$

and r_2 can be given by the following eguation

$$E_{th} = \frac{2\lambda}{vr_2}\tan^{-1}\left(\frac{\delta}{2r_2}\right).$$

According to [1], to cover an area A, the number of nodes required would be of the order $O\left(\frac{A}{r^2}\right)$, where r is the sensing radius lying between r_1 and r_2. Via simulations, it is shown that using r_2 is a good estimation for finding the number of sensors required and the probability of detection is 98% or above.

2.4 Coverage and Connectivity

In this section, we discuss some works that consider the coverage and connectivity of sensor networks [9, 28, 35, 43]. Each sensor is assumed to have a fixed sensing region and a fixed communication range, both of which are modeled as disks. The goal is to achieve certain sensing coverage and/or communication connectivity requirements.

For the sensor network to operate successfully, the active nodes must maintain both sensing coverage and network connectivity. Reference [35] proposes another solution to determine if a target area is k-covered and further studies the relationship between coverage and connectivity. To determine the coverage level, this work looks at how intersection points between sensors' sensing ranges are covered. It claims that a region is k-covered by a set of sensors if all intersection points between sensors and between any sensor and the boundary of this region are at least k-covered. For the network communication connectivity, it claims that if a region is k-covered, then the sensor network is k-connected as long as those sensors' communication ranges are no less than twice their sensing ranges.

Based on the above two properties, a *Coverage Configuration Protocol (CCP)* that can provide different degrees of coverage and meanwhile maintain communication connectivity is presented in [35] when the communication ranges are no less than twice their sensing ranges.

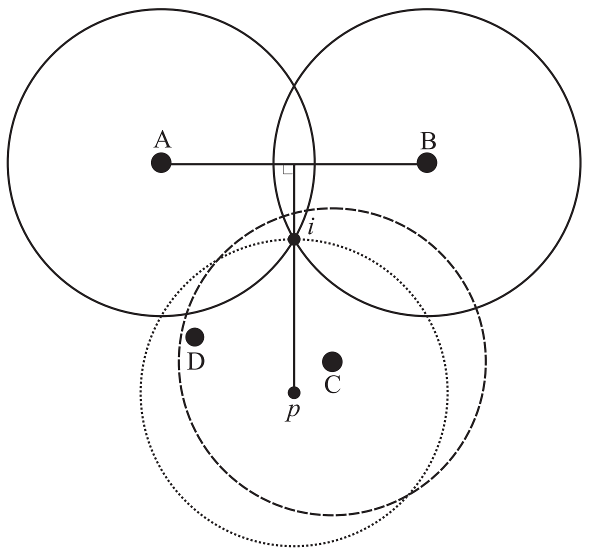


Figure 2.7: An example of the OGDC algorithm.

Initially, all sensors are in the *active* state. If an area exceeds the required degree of coverage, redundant nodes will find themselves unnecessary and switch to the *sleep* state. A sensor is unnecessary to stay active if all the intersection points inside its sensing circle are at least k-covered. A sleeping node also periodically wakes up and enters the *listen* state. In the listen state, the sensor evaluates whether it is necessary to return to the active state.

If the communication ranges are less than twice the sensing ranges, reference [35] proposes to integrate CCP with SPAN [7] to provide both sensing coverage and communication connectivity. SPAN is a connectivity-maintaining protocol which can turn off unnecessary nodes such that all active nodes are connected through a communication backbone and all inactive nodes are directly connected to at least one active node. Reference [35] proposes that an inactive node should become active following rules of SPAN or CCP. An active node will turn to sleep if it satisfies neither SPAN's nor CCP's wakeup rules.

How to maintain the sensing coverage and connectivity is also addressed in [43]. Similar to [35], the paper also shows that coverage can imply connectivity if the transmission range is at least twice of the sensing range. If so, we only need to focus on the coverage problem. A decentralized density control algorithm called *Optimal Geographical Density Control (OGDC)* is proposed to choose as few number of working nodes as possible to cover the network. Initially, all nodes are in the *UNDECIDED* state. We first find several starting nodes to enter the *ON* state. Nodes in the *ON* state may bring other *UNDECIDED*-state nodes to the *ON* state. The basic idea is to reduce the overlapping areas that are covered by nodes in the *ON* state.

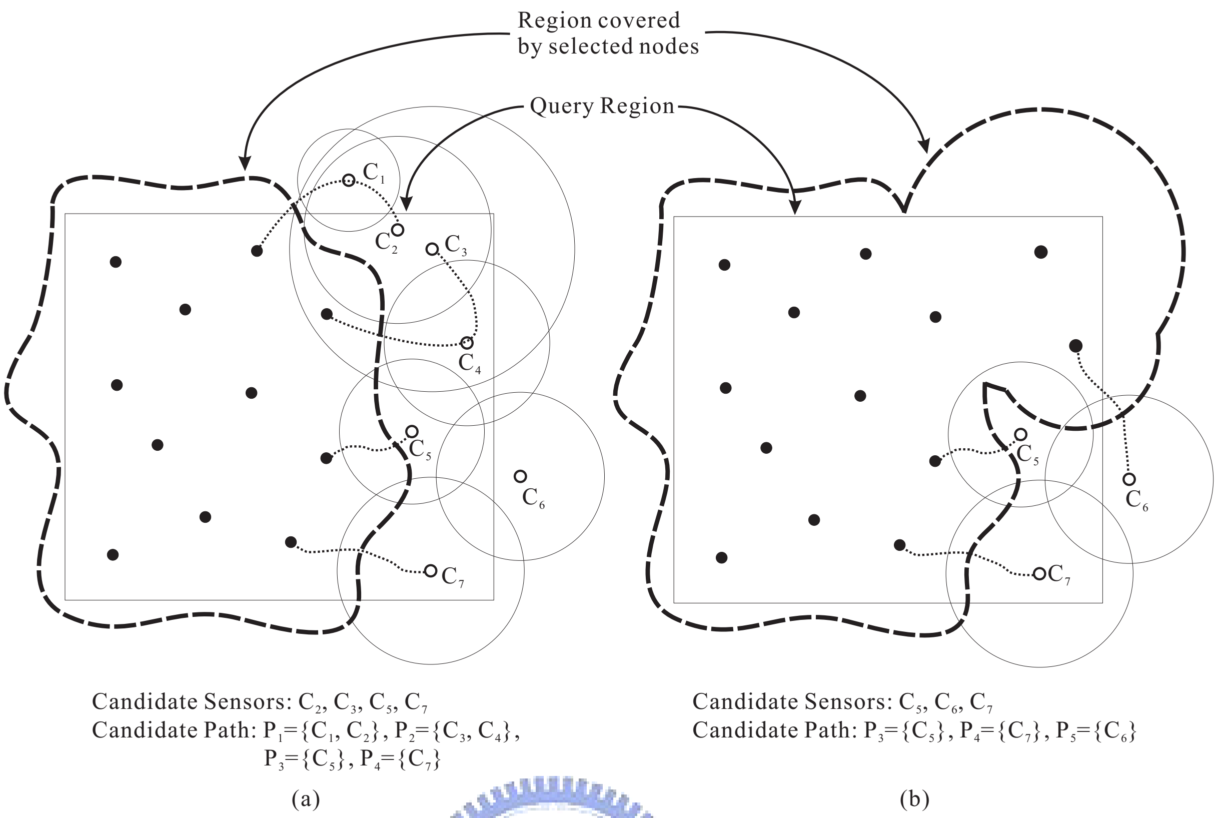
For example, in Fig. 2.7, there are four sensors $\{A, B, C, D\}$ and A is a starting node. Then A selects its neighbor B to enter the *ON* state because B's distance from A is closest to $\sqrt{3}r$, where r is the sensing radius of each sensor. To cover the intersection point i of A's and B's circles, we then select the node whose position is closest to the optimal position p which is on the perpendicular bisector of the line connecting A and B and is at a distance of r from i. As a result, C is selected and turned to the *ON* state. This procedure is repeated until the whole network has been covered. Note that a node in the *UNDECIDED* state can enter the *OFF* state if it finds its sensing range has been fully covered by other *ON*-state nodes.

Reference [28] investigates three coverage-related issues about a sensor network based on the grid-based deployment. In a unit square, $\sqrt{n} \times \sqrt{n}$ sensors are deployed in the field. However, each sensor has a probability of $p(n)$ to remain functioning, and a probability of $1 - p(n)$ to be dead. The authors show that when $p(n)r^2(n) \approx \frac{\log(n)}{n}$, it is very likely that the network will remain fully covered and connected, where $r(n)$ is the sensing and communication range of each sensor. Also, under such a condition, the network diammeter will be of the order $\sqrt{\frac{n}{\log(n)}}$.

Reference [9] investigates the coverage and connectivity issues from another point of view. When a spatial query is issued to the sensor network to request the data of interest in a geographical region, we may like to select the smallest subset of sensors which are connected and are sufficient to cover the region. The proposed solution is a greedy algorithm which recurrently selects a path of sensors that is connected to an already selected sensor and then adds these sensors into the selected subset until the given query region is completely covered. The greedy rule of the algorithm is to select a path of sensors who can cover the largest uncovered query region at each stage. Fig. 2.8 shows an example with two consecutive stages of the algorithm. Fig. 2.8(b) is resulted from (a) by selecting sensors of path P_2 since P_2 consists of sensors C_3 and C_4 who together cover the largest uncovered region.

2.5 Coverage-Preserving and Energy-Conserving Protocols

Since sensors are usually powered by batteries, sensors' on-duty time should be properly scheduled to conserve energy. If some nodes share the common sensing region and task, then we can turn off some of them to conserve energy and thus extend the lifetime of the network. This is feasible if turning off such a node still provides the same "coverage" (i.e., the provided coverage is not affected). An example is shown in Fig. 2.9(a). The sensor e can be put into sleeping mode



Region covered by selected nodes
Query Region
C_1
C_2
C_3
C_4
C_5
C_6
C_7
Candidate Sensors: C_2, C_3, C_5, C_7
Candidate Path: $P_1=\{C_1, C_2\}$, $P_2=\{C_3, C_4\}$, $P_3=\{C_5\}$, $P_4=\{C_7\}$
(a)
Candidate Sensors: C_5, C_6, C_7
Candidate Path: $P_3=\{C_5\}$, $P_4=\{C_7\}$, $P_5=\{C_6\}$
(b)

Figure 2.8: An example of the progress of the algorithm in [9]. Dotted lines show the connectivity between sensors.

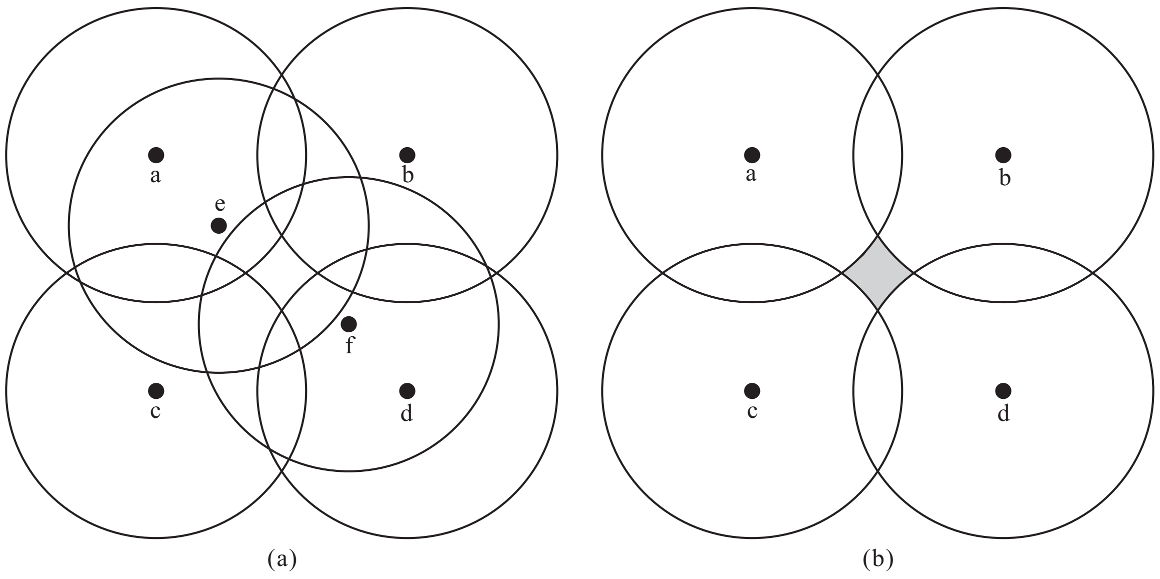


Figure 2.9: An example of the blind point if both sensors e and f are put into sleeping at the same time.

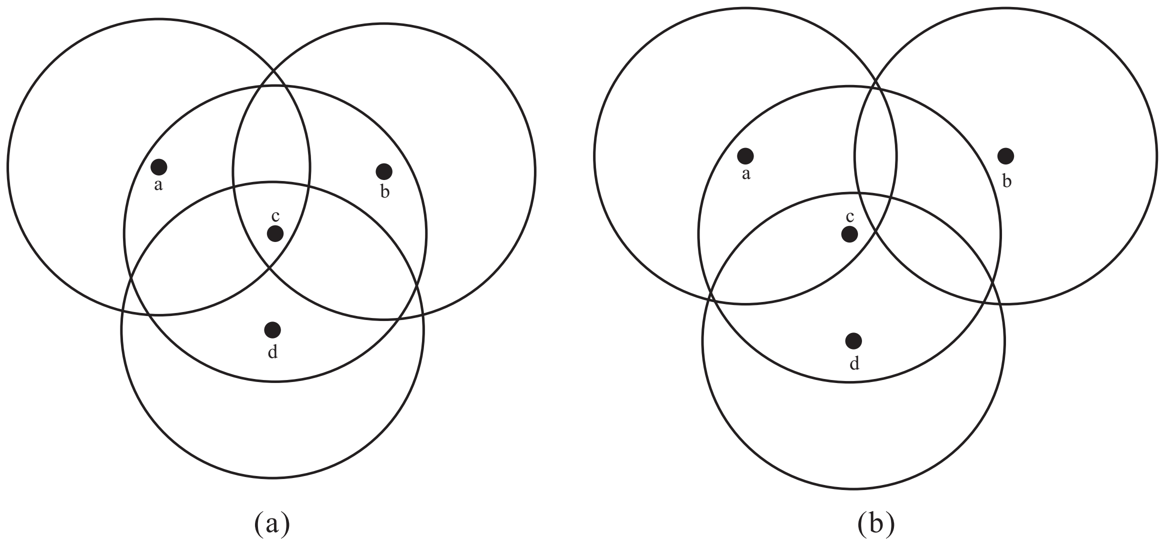


Figure 2.10: Examples that (a) sensor c satisfies the off-duty eligibility rule of [32] and (b) sensor c does not satisfy the off-duty eligibility rule of [32].

since all its sensing area is covered by the other nodes. Sensor f satisfies this condition too and can go to sleeping mode. However, e and f are not allowed to be turned off at the same time; otherwise a blind point, which is a region not covered by any sensor, could appear, as shown in Fig. 2.9(b). As a result, sensors not only need to be check if they satisfy certain eligibility rules but also need to be carefully scheduled.

Reference [30] proposes a heuristic to select mutually exclusive sets of sensor nodes such that each set of sensors can provide a complete coverage of the monitored area. They claim that this problem is a NP-complete problem by it reducing to the minimum cover problem. The key idea of the proposed heuristic is to find out which sensors cover fields that are less covered by other sensors and then avoid including those sensors into the same set. Also targeted at turning off some redundant nodes, [41] proposes a probe-based density control algorithm to put some nodes in a sensor-dense area to a doze mode to ensure a long-lived, robust sensing coverage. In this solution, nodes are initially in the sleeping mode. After a sleeping node wakes up, it broadcasts a probing message within a certain range and then waits for a reply. If no reply is received within a pre-defined time period, it will keep active until it depletes its energy. The coverage degree (density) is controlled by sensor's probing range and wake-up rate. However, this probing-based approach has no guarantee of sensing coverage and thus blind points could appear.

A coverage-preserving node scheduling scheme is presented in [32] to determine when a node can be turned off and when it should be rescheduled to become active again. It is based on an eligibility rule which allows a node to turn itself off as long as other neighboring nodes can cover its sensing area. After evaluating its eligibility for off-duty, each sensor adopts a back-off scheme to prevent the appearance of blind points. If a node is eligible for off-duty, it will delay a random back-off time before actually turning itself off. During this period of time, if it receives any message from its neighbors requesting to go to sleep, it marks the sender as an off-duty node and reevaluates its eligibility. If the eligibility still holds after the back-off time, this node broadcasts a message to inform its neighbors, waits for a short period of time, and then actually turns itself off. A sleeping node will periodically wake up to check if it is still eligible for off-duty and then decide to keep sleeping or go back to on-duty.

However, the solution in [32] may lead to excess energy consumption. An example is shown in Fig. 2.10. Based on the eligibility rule proposed in [32], a sensor only regards a node whose sensing range can cover the sensor as a neighboring node. In Fig. 2.10(a), sensor c is eligible for off-duty since its sensing region is covered by its neighboring nodes a, b and d. In contrast to the above case, in Fig. 2.10(b), sensor c is not eligible for off-duty since sensor b is not regarded as a neighboring node of c. According to the eligibility rule of [32], c cannot be turned off. In fact, c's sensing region is fully covered by sensors a, b and d, thus leading to excess energy consumption.

Another node scheduling scheme is proposed in [40]. In this scheme, the time axis is divided into rounds with equal duration. Each sensor node randomly generates a reference time in each round. In addition, the whole sensing area is divided into grid points which are used to evaluate whether the area is sufficiently covered or not. Each sensor has to join the schedule of each grid point covered by itself based on its reference time such that the grid point is covered by at least one sensor at any moment of a round. Then a sensor's on-duty time in each round is the union of schedules of grid points covered by the sensor. However, this scheme may suffer from the time synchronization problem in a large-scale sensor network.

A coverage-aware self-scheduling scheme based on probabilistic sensing model is proposed in [17]. Each sensor S_j is assumed to be able to detect a nearby event happened at location P_i with a probability

$$S_j(P_i) = \frac{1}{(1 + \alpha D_{ij})^{\beta}},$$

where D_{ij} is the distance between sensor S_j and point P_i and constants α and β are device-depended parameters. Thus, the level of coverage perceived by P_i contributed by all sensors can be written as

$$C(P_i) = 1 - \prod_{\forall S_j}(1 - S_j(P_i)).$$

Now suppose a sensor S_n is removed from the network. The loss of coverage at point P_i can be derived by

$$\Delta(P_i) = S_n(P_i) \prod_{\forall S_j \neq S_n} (1 - S_j(P_i)).$$

Therefore, sensor S_n's contribution to the network coverage can be defined by summing the losses over all possible points

$$SD_n = \sum_{\forall P_i \text{ within distance } R \text{ from } S_n} \Delta(P_i),$$

where R is the largest range that a sensor can detect with a predefined accuracy.

A self-scheduling scheme based on above SD value is then proposed in [17]. Periodically, each sensor S_i calculates its SD value in a distributed manner and decides whether to enter sleeping state with a hibernating probability defined as follows

$$P_{hibernate} = (SD_{base} - SD_i)/SD_{base},$$

where SD_{base} is half of the maximum possible SD value. Therefore, a sensor with a higher SD value has a higher chance to stay active. To prevent an area becoming uncovered due to all sensors in this area turning themselves off at the same time, a random backoff mechanism as [32] is used.

Chapter 3

The Coverage Problem in a Two-Dimensional Space

In this chapter, we study the coverage problem in a two-dimensional space. We formulate this problem as a decision problem, whose goal is to determine whether every point in the service area of the sensor network is covered by at least k sensors, where k is a given parameter. The sensing ranges of sensors can be unit disks or non-unit disks. We present polynomial-time algorithms, in terms of the number of sensors, that can be easily translated to distributed protocols. The result is a generalization of some earlier results where only $k = 1$ is assumed. Applications of the result include determining insufficiently covered areas in a sensor network, enhancing fault-tolerant capability in hostile regions, and conserving energies of redundant sensors in a randomly deployed network.

3.1 Problem Statement

We are given a set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, in a two-dimensional area A. Each sensor $s_i, i = 1..n$, is located at coordinate (x_i, y_i) inside A and has a sensing range of r_i, i.e., it can monitor any point that is within a distance of r_i from s_i.

Definition 1 A location in A is said to be *covered* by s_i if it is within s_i's sensing range. A location in A is said to be j*-covered* if it is within at least j sensors' sensing ranges.

Definition 2 A *sub-region* in A is a set of points who are covered by the same set of sensors.

We consider two versions of the coverage problem as follows.

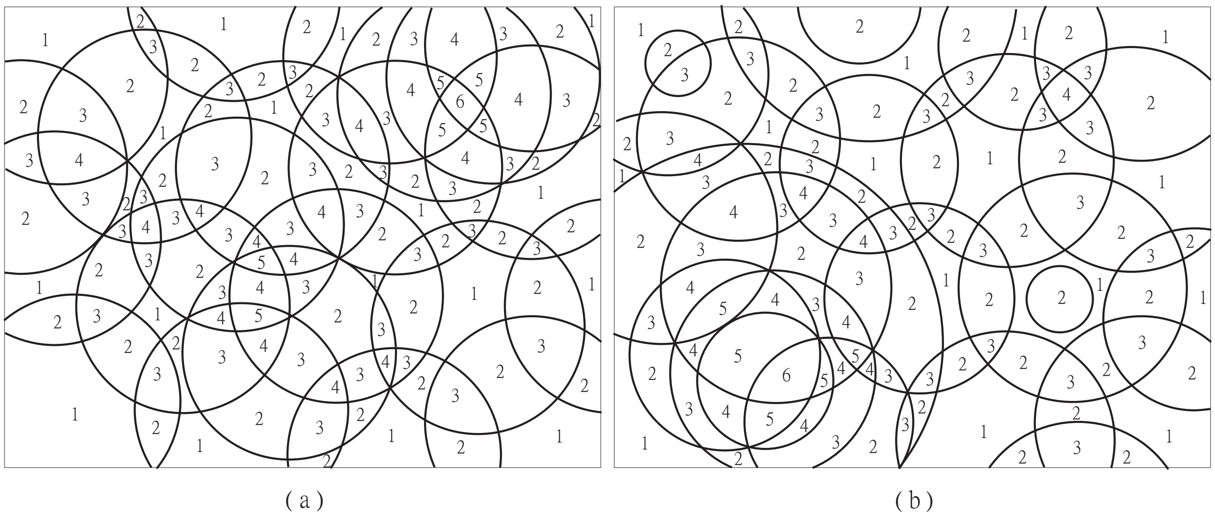


Figure 3.1: Examples of the coverage problem: (a) the sensing ranges are unit disks, and (b) the sensing ranges are non-unit disks. The number in each sub-region is its coverage.

Definition 3 Given a natural number k, the *k-Non-unit-disk Coverage (k-NC) Problem* is a decision problem whose goal is to determine whether all points in A are k-covered or not.

Definition 4 Given a natural number k, the *k-Unit-disk Coverage (k-UC) Problem* is a decision problem whose goal is to determine whether all points in A are k-covered or not, subject to the constraint that $r_1 = r_2 = \cdots = r_n$.

3.2 The Proposed Solutions

At the first glance, the coverage problem seems to be very difficult. One naive solution is to find out all sub-regions divided by the sensing boundaries of all n sensors (i.e., n circles), and then check if each sub-region is k-covered or not, as shown in Fig. 3.1. Managing all sub-regions could be a difficult and computationally expensive job in geometry. There may exit as many as $O(n^2)$ sub-regions divided by the circles. Also, it may be difficult to calculate these sub-regions.

3.2.1 The k-UC Problem

In the section, we propose a solution to the k-UC problem, which has a cost of $O(nd \log d)$, where d is the maximum number of sensors whose sensing ranges may intersect a sensor's sensing range. Instead of determining the coverage of each sub-region, our approach tries to

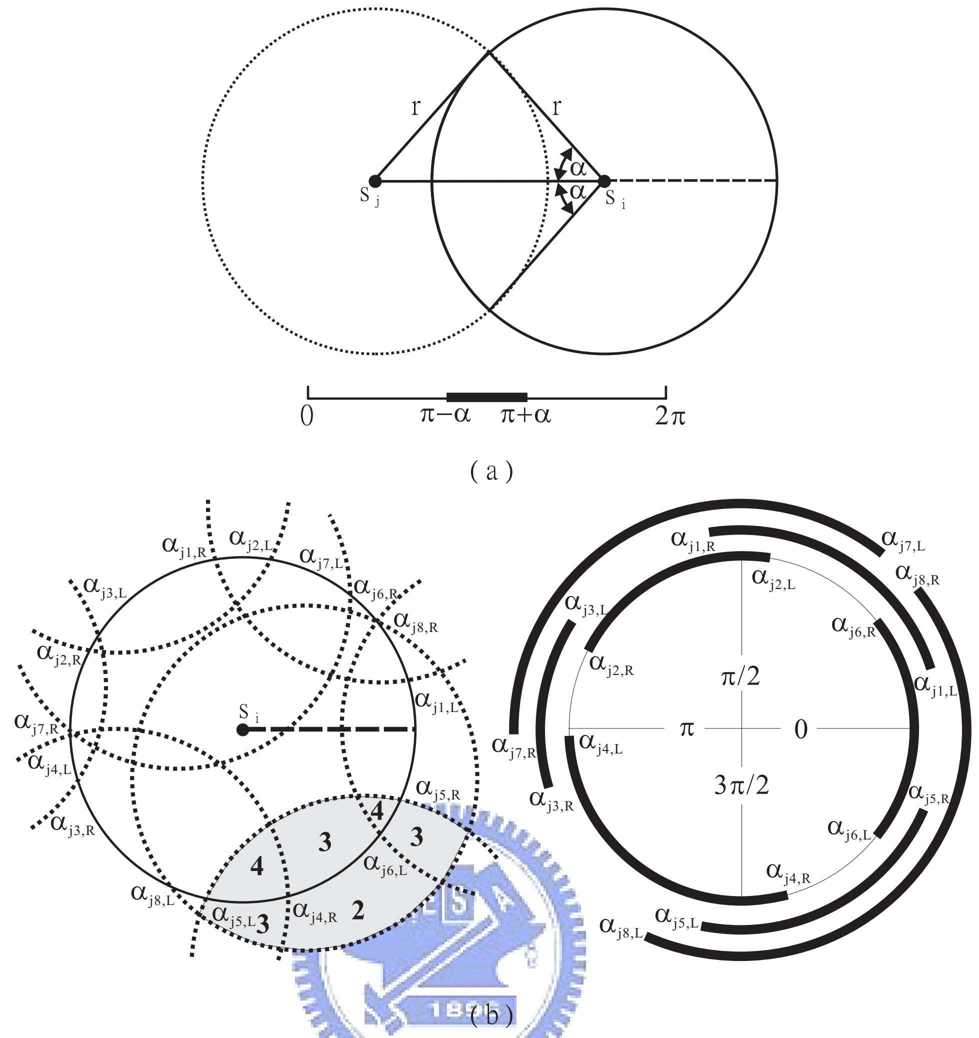


Figure 3.2: (a) Determining the segment of s_i's perimeter covered by s_j, and (b) determining the perimeter-coverage of s_i's perimeter.

look at how the perimeter of each sensor's sensing range is covered. Specifically, our algorithm tries to determine whether the perimeter of a sensor under consideration is sufficiently covered. By collecting this information from all sensors, a correct answer can be obtained.

Definition 5 Consider any two sensors s_i and s_j. A point on the perimeter of s_i is *perimeter-covered by* s_j if this point is within the sensing range of s_j.

Definition 6 Consider any sensor s_i. We say that s_i is k*-perimeter-covered* if all points on the perimeter of s_i are perimeter-covered by at least k sensors other than s_i itself. Similarly, a segment of s_i's perimeter is k*-perimeter-covered* if all points on the segment are perimeter-covered by at least k sensors other than s_i itself.

Below, we propose an $O(d \log d)$ algorithm to determine whether a sensor is k-perimeter-covered or not. Consider two sensors s_i and s_j located in positions (x_i, y_i) and (x_j, y_j), respectively. Denote by $d(s_i, s_j) = \sqrt{|x_i - x_j|^2 + |y_i - y_j|^2}$ the distance between s_i and s_j. If $d(s_i, s_j) > 2r$, then s_j does not contribute any coverage to s_i's perimeter. Otherwise, the range of perimeter of s_i covered by s_j can be calculated as follows (refer to the illustration in Fig. 3.2(a)). Without loss of generality, let s_j be resident on the west of s_i (i.e., $y_i = y_j$ and $x_i > x_j$). The angle $\alpha = \arccos(\frac{d(s_i,s_j)}{2r})$. So the arch of s_i falling in the angle $[\pi - \alpha, \pi + \alpha]$ is perimeter-covered by s_j.

The algorithm to determine the perimeter coverage of s_i works as follows.

1. For each sensor s_j such that $d(s_i, s_j) \leq 2r$, determine the angle of $s_i's$ arch, denoted by $[\alpha_{j,L}, \alpha_{j,R}]$, that is perimeter-covered by s_j.

2. For each neighboring sensor s_j of s_i such that $d(s_i, s_j) < 2r$, place the points $\alpha_{j,L}$ and $\alpha_{j,R}$ on the line segment $[0, 2\pi]$, and then sort all these points in an ascending order into a list L. Also, properly mark each point as a left or right boundary of a coverage range, as shown in Fig. 3.2(b).

3. (Sketched) Traverse the line segment $[0, 2\pi]$ by visiting each element in the sorted list L from left to right and determine the perimeter-coverage of s_i.

The above algorithm can determine the coverage of each sensor's perimeter efficiently. Below, we relate the perimeter-coverage property of sensors to the coverage property of the network area.

Lemma 1 *Suppose that no two sensors are located in the same location. Consider any segment of a sensor s_i that divides two sub-regions in the network area A. If this segment is k-perimeter-covered, the sub-region that is outside s_i's sensing range is k-covered and the sub-region that is inside s_i's sensing range is $(k+1)$-covered.*

Proof. The proof is directly from Definition 6. Since the segment is k-perimeter-covered, the sub-region outside s_i's sensing range is also k-covered due to the continuity of the sub-region. The sub-region inside s_i's sensing range is $(k+1)$-covered because it is also covered by s_i. □

An example is demonstrated in Fig. 3.2(b). The gray areas in Fig. 3.2(b) illustrate how the above lemma works .

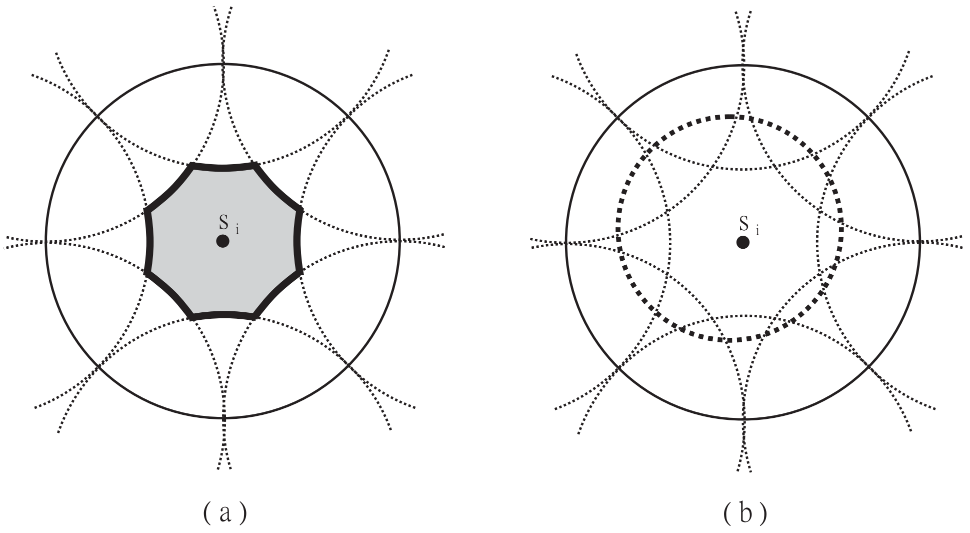


Figure 3.3: Some examples to utilize the result in Theorem 1.

Theorem 1 *Suppose that no two sensors are located in the same location. The whole network area A is k-covered iff each sensor in the network is k-perimeter-covered.*

Proof. For the "if" part, observe that each sub-region inside A is bounded by at least one segment of a sensor s_i's perimeter. Since s_i is k-perimeter-covered, by Lemma 1, this sub-region is either k-covered or $(k+1)$-covered, which proves the "if" part.

For the "only if" part, it is clear by definition that for any segment of a sensor s_i's perimeter that divides two sub-regions, both these sub-regions are at least k-covered. Further, observe that the sub-region that is inside s_i's sensing range must be covered by one more sensor, s_i, and is thus at least $(k+1)$-covered. So excluding s_i itself, this segment is perimeter-covered by at least k sensors other than s_i itself, which proves the "only if" part. □

Note that Theorem 1 is true when *all* sensors are claimed to be k-perimeter-covered. When a specific sensor s_i is k-perimeter-covered, it only guarantees that each point right outside s_i's perimeter is k-covered, and each point right inside s_i's perimeter is $(k+1)$-covered. However, it does not guarantee that *all* points inside s_i's perimeter is $(k+1)$-covered. An example is shown in Fig. 3.3. In Fig. 3.3(a), sensor s_i is 2-perimeter-covered since each segment of its perimeter is covered by two sensors. This only implies the coverage levels of the points nearby the perimeter of s_i. The gray area, which is outside the coverage of s_i's neighboring sensors, is only 1-covered. In fact, the segments that bound the gray area are only 1-perimeter-covered. If we add another sensor to cover these segments (shown in thick dotted line) as shown in Fig. 3.3(b), then s_i's sensing region will be 2-covered.

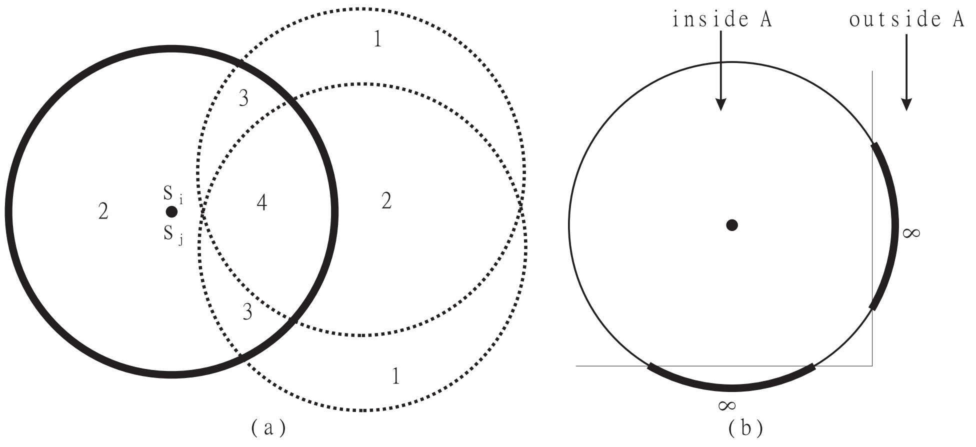


Figure 3.4: Some special cases: (a) two sensors falling in the same location (the number in each sub-region is its level of coverage), and (b) the sensing range of a sensor exceeding the network area A.

Below, we comment on several special cases which we leave unaddressed on purpose for simplicity in the above discussion. When two sensors s_i and s_j fall in exactly the same location, Lemma 1 will not work because for any segment of s_i and s_j that divides two sub-regions in the network area, a point right inside s_i's and s_j's sensing ranges and a point right outside their sensing ranges will differ in their coverage levels by two, making Lemma 1 incorrect (refer to the illustration in Fig. 3.4(a)). Other than this case, all neighboring sub-regions in the network will differ in their coverage levels by exactly one. Since in most applications we are interested in areas that are insufficiently covered, one simple remedy to this problem is to just ignore one of the sensors if both sensors fall in exactly the same location. Another solution is to first run our algorithm by ignoring one sensor, and then increase the coverage levels of the sub-regions falling in the ignored sensor's range by one afterward. The other boundary case is that some sensors' sensing ranges may exceed the network area A. In this case, we can simply assign the segments falling outside A as ∞-perimeter-covered, as shown in Fig. 3.4(b).

3.2.2 The k-NC Problem

For the non-unit-disk coverage problem, sensors' sensing ranges could be different. However, most of the results derived above remain the same. Below, we summarize how the k-NC problem is solved.

First, we need to define the how the perimeter of a sensor's sensing range is covered by other sensors. Consider two sensors s_i and s_j located in positions (x_i, y_i) and (x_j, y_j) with sensing

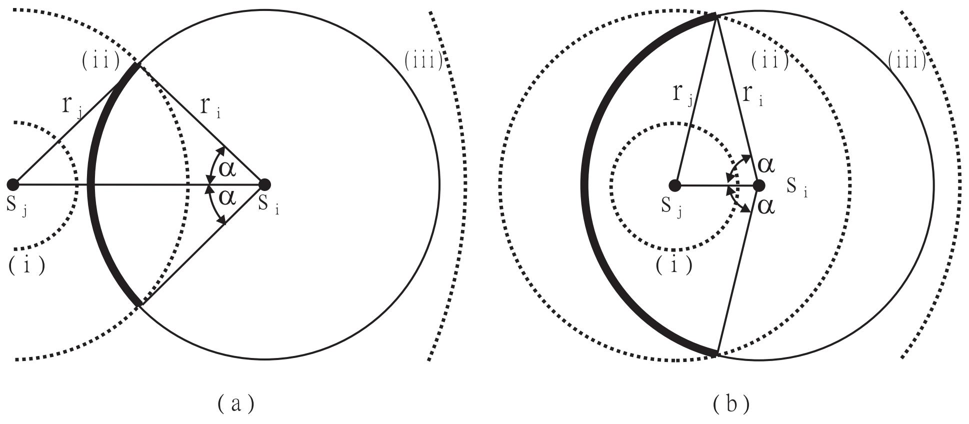


Figure 3.5: The coverage relation of two sensors with different sensing ranges: (a) s_j not in the range of s_i, and (b) s_j in the range of s_i.

ranges r_i and r_j, respectively. Again, without loss of generality, let s_j be resident on the west of s_i. We address how s_i is perimeter-covered by s_j. There are two cases to be considered.

Case 1: Sensor s_j is outside the sensing range of s_i, i.e., $d(s_i, s_j) > r_i$.

(i) If $r_j < d(s_i, s_j) - r_i$, then s_i is not perimeter-covered by s_j.

(ii) If $d(s_i, s_j) - r_i \leq r_j \leq d(s_i, s_j) + r_i$, then the arch of s_i falling in the angle $[\pi - \alpha, \pi + \alpha]$ is perimeter-covered by s_j, where α can be derived from the formula:

$$r_j^2 = r_i^2 + d(s_i, s_j)^2 - 2r_i \cdot d(s_i, s_j) \cdot \cos(\alpha). \tag{3.1}$$

(iii) If $r_j > d(s_i, s_j) + r_i$, then the whole range $[0, 2\pi]$ of s_i is perimeter-covered by s_j.

Case 2: Sensor s_j is inside the sensing range of s_i, i.e., $d(s_i, s_j) \leq r_i$.

(i) If $r_j < r_i - d(s_i, s_j)$, then s_i is not perimeter-covered by s_j.

(ii) If $r_i - d(s_i, s_j) \leq r_j \leq r_i + d(s_i, s_j)$, then the arch of s_i falling in the angle $[\pi - \alpha, \pi + \alpha]$ is perimeter-covered by s_j, where α is as defined in Eq. (3.1).

(iii) If $r_j > d(s_i, s_j) + r_i$, then the whole range $[0, 2\pi]$ of s_i is perimeter-covered by s_j.

The above cases are illustrated in Fig. 3.5. Based on such classification, the same algorithm to determine the perimeter coverage of a sensor can be used. Lemma 1 and Theorem 1 still hold true (observe that in the corresponding proofs, we do not use any property about the absolute sensing ranges of sensors).

3.2.3 Complexity Analysis

Consider the algorithm in Section 3.2.1. Let d be the maximum number of sensors that are neighboring to a sensor ($d \leq n$). The complexities of steps 1 and 2 are $O(d)$ and $O(d \log d)$, respectively. The last step 3, though sketched, can be easily implemented as follows. Whenever an element $\alpha_{j,L}$ is traversed, the level of perimeter-coverage should be increased by one. Whenever an element $\alpha_{j,R}$ is traversed, the level of perimeter-coverage should be decreased by one. Since the sorted list L will divide the line segment $[0, 2\pi]$ into as many as $2d+1$ segments, the complexity of step 3 is $O(d)$. So the complexity to determine a sensor's perimeter coverage is $O(d \log d)$. The overall complexity for the k-UC problem is thus $O(nd \log d)$. The k-NC problem can also be solved with complexity $O(nd \log d)$, except that the neighbors of a sensor need to be redefined. The work [35] also proposes a solution to determine the coverage level of a sensor network. It looks at how intersection points between sensors' sensing ranges are covered. Since there are as many as $O(n^2)$ intersection points in the network and the calculation of the coverage level of each intersection point takes time $O(n)$, the overall complexity is $O(n^3)$.

3.3 Simulation Results and a Sensor Coverage Toolkit

We have developed a simulator and implemented a toolkit based on the proposed algorithms. Square sensor fields are simulated with randomly placed nodes. There are two settings of sensing ranges: unit-disc sensing range and non-unit-disc sensing range. All results presented below are from the average of at least 1000 runs.

First, we investigate the level of coverage (i.e., k) that can be achieved by using different numbers of sensors. Sensor fields of sizes 500×500 and 1000×1000 are simulated with $100 \sim 1000$ nodes. The unit-disc sensing range is 100 units and the non-unit-disc sensing range falling uniformly between $50 \sim 150$ units. Both the average and the maximum levels of coverage are evaluated. The results are in Fig. 3.6. As can be seen, the average value of k grows about linearly as the number of sensors increases.

Next, we investigate the level of coverage that can be achieved by setting different sensing ranges of sensors. Sensor fields of sizes 500×500 and 1000×1000 are simulated with 500 nodes. For the unit-disc case, the sensing range is fixed from 50 to 150 units. For the non-unit-disc case, we first pick an average sensing range avg, and the sensors' sensing ranges are uniformly distributed between $avg - 50$ and $avg + 50$. The results are in Fig. 3.7. The average

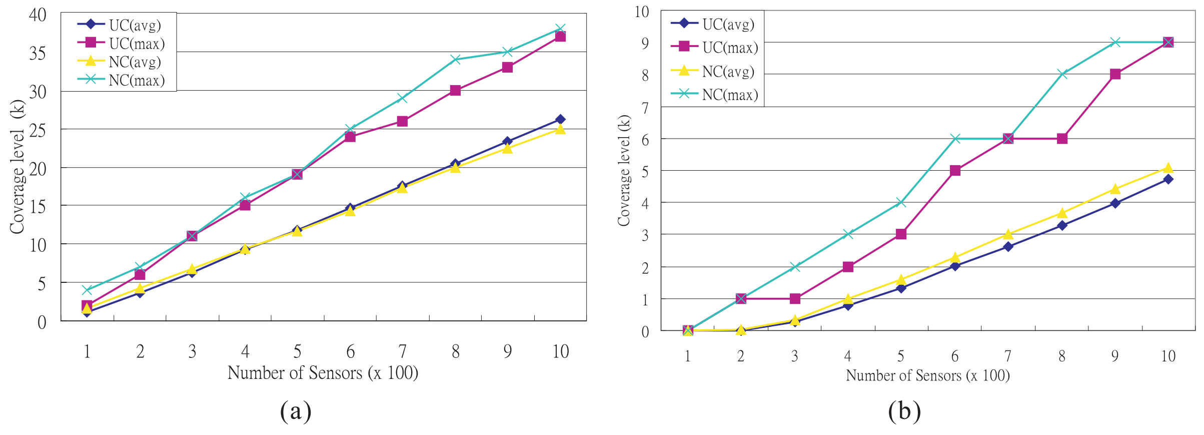


Figure 3.6: Number of sensors v.s. coverage level for sensor fields of sizes: (a) 500×500 and (b) 1000×1000.

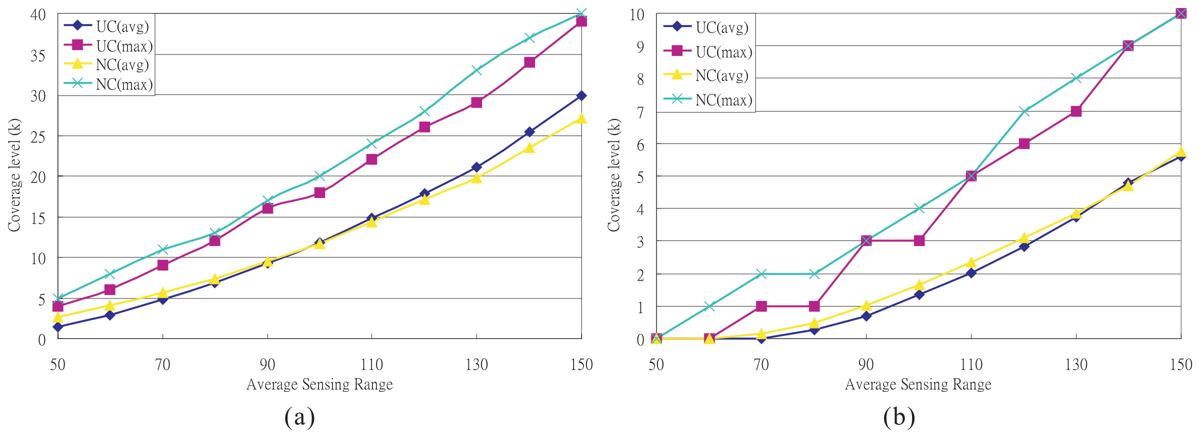


Figure 3.7: Sensing range v.s. coverage level for sensor fields of sizes: (a) 500×500 and (b) 1000×1000.

value of k grows as the average sensing range of sensors increases.

We have also implemented a toolkit based on the proposed algorithms to determine the coverage level of a given sensing field. Fig. 3.8 shows the user interface of the toolkit. In the drawing area, one can easily deploy sensors by pointing out their locations and dragging their sensing ranges. By clicking on the “Deploy” button, the deployment of sensors will be fed into our program. There are three major functions of this toolkit, as described below.

1. *Compute the Level of Coverage:* By clicking on the “Compute Coverage” button and then the “Display Coverage” button, the system will calculate and return the current coverage level of the whole area, as illustrated in Fig. 3.9(a).

2. *Color the Drawing Area:* By clicking on the “Paint the drawing area” button, the drawing area will be colored based on each region’s coverage level. The coloring speed can also be modified, which will reflect on the coloring quality. An example is shown in Fig. 3.9(b).

3. *Display Insufficiently Covered Segments:* One can first select the desired value of k followed by clicking on the “Commit” button to feed k into the system. Clicking on the “Get Low Coverage Segments” button will generate an output file which contains all segments that are insufficiently k-perimeter-covered, as shown in the Fig. 3.10. Each line in the file is a segment of one sensor’s perimeter that is insufficiently covered. Fields in a line include: sensor ID, location, sensing range, starting and ending angles of the corresponding segment, and the levels of coverage inside and outside this segment.

This toolkit is publicly downloadable from http://hscc.csie.nctu.edu.tw/download/coverage.zip.

3.4 Applications and Extensions of the Coverage Problem

The sensor coverage problem, although modeled as a decision problem, can be extended further in several ways for many interesting applications. The proposed results can also be extended for more realistic situations. In the following, we suggest several applications of the coverage problem and possible extensions of our results.

3.4.1 Discovering Insufficiently Covered Regions

For a sensor network, one basic question is whether the network area is fully covered. Our modeling of the k-UC and k-NC problems can solve the sensor coverage problem in a more

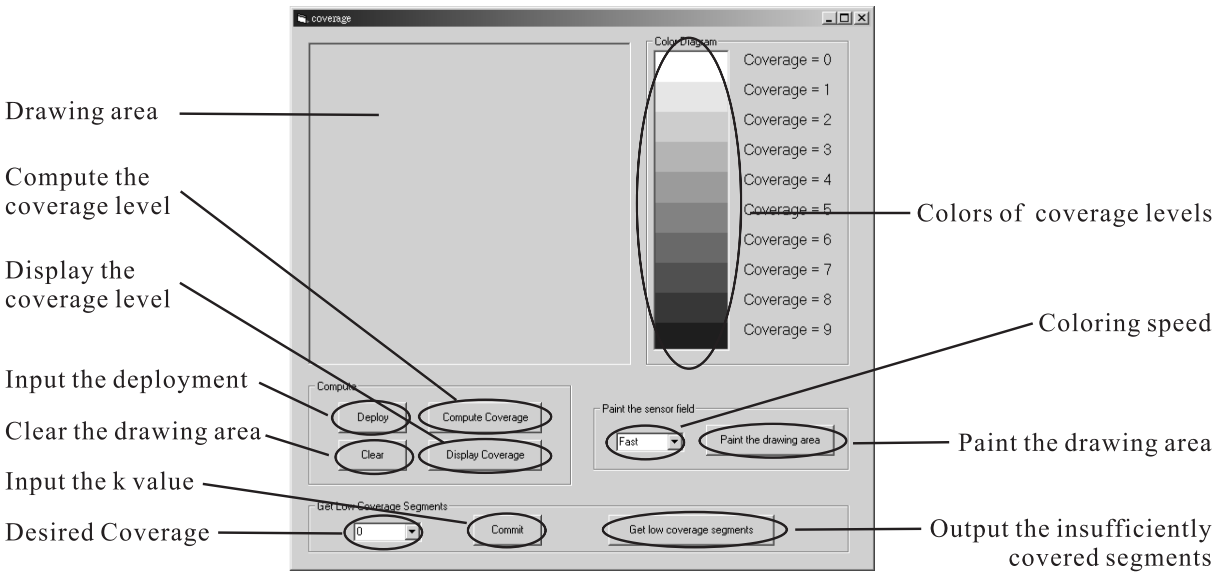


Figure 3.8: Functional descriptions of the toolkit.

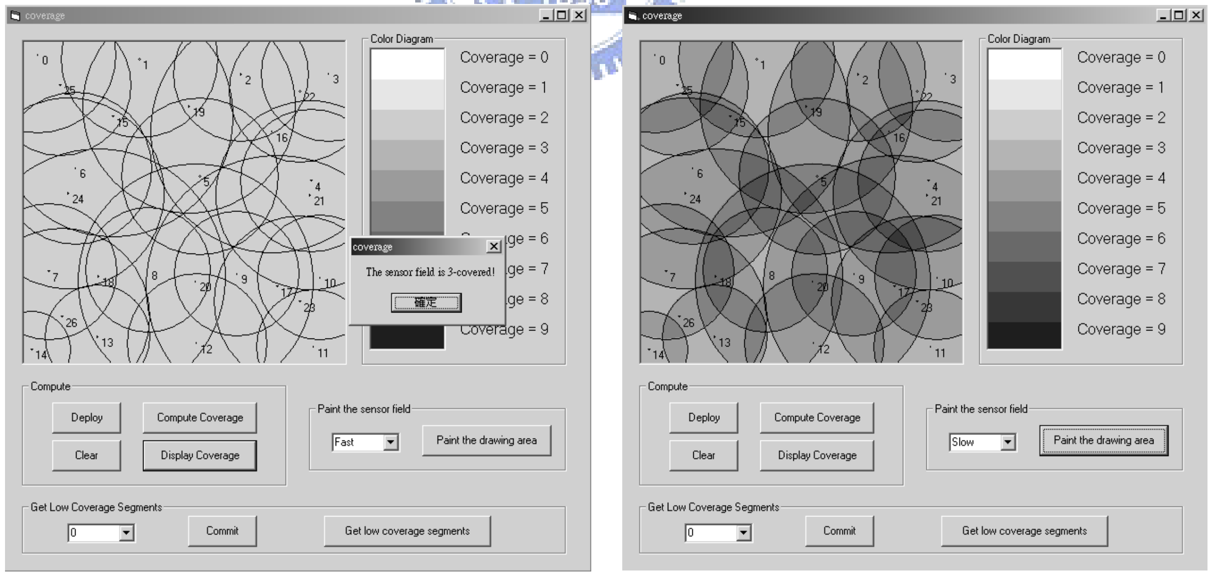
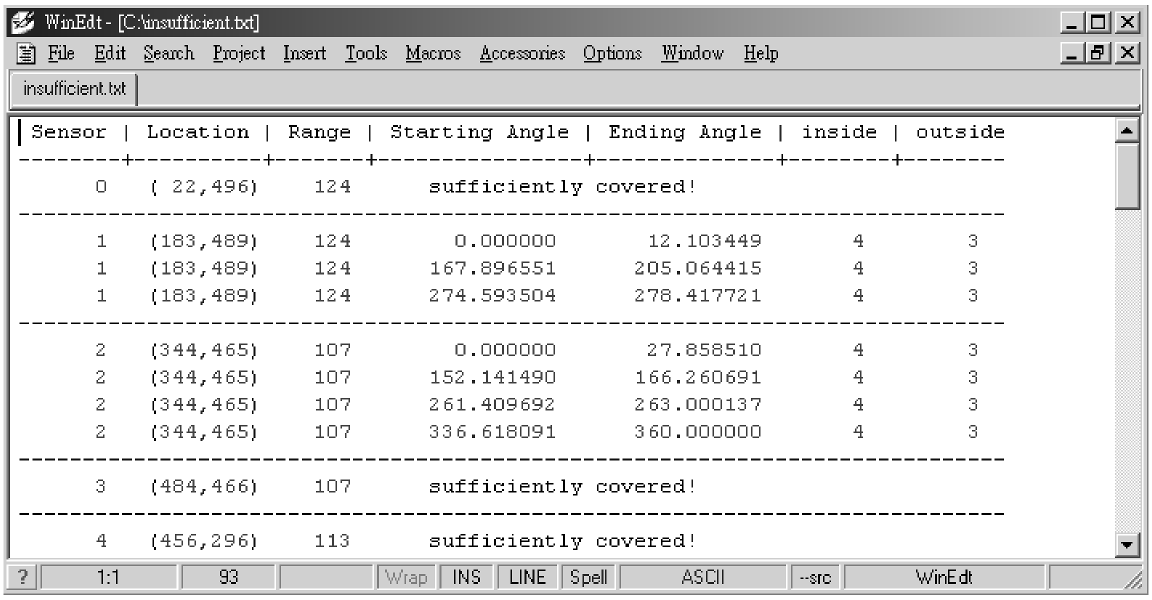


Figure 3.9: Execution results of the toolkit: (a) coverage level and (b) painting results.


WinEdt - [C:\insufficient.txt]
File Edit Search Project Insert Tools Macros Accessories Options Window Help
insufficient.txt
? 1:1 93 Wrap INS LINE Spell ASCII --src WinEdt

```
| Sensor | Location | Range | Starting Angle | Ending Angle | inside | outside
--------+----------+-------+----------------+--------------+--------+--------
      0   ( 22,496)    124      sufficiently covered!
--------------------------------------------------------------------------------
      1   (183,489)    124       0.000000       12.103449       4        3
      1   (183,489)    124     167.896551      205.064415       4        3
      1   (183,489)    124     274.593504      278.417721       4        3
--------------------------------------------------------------------------------
      2   (344,465)    107       0.000000       27.858510       4        3
      2   (344,465)    107     152.141490      166.260691       4        3
      2   (344,465)    107     261.409692      263.000137       4        3
      2   (344,465)    107     336.618091      360.000000       4        3
--------------------------------------------------------------------------------
      3   (484,466)    107      sufficiently covered!
--------------------------------------------------------------------------------
      4   (456,296)    113      sufficiently covered!
```

Figure 3.10: Insufficiently 4-perimeter-covered segments for the example in Fig. 3.9.

general sense by determining if the network area is k-covered or not. A larger k can support a more fine-grained sensibility. For example, if $k = 1$, we can only detect in which sensor an event has happened. Using a larger k, the location of the event can be reduced to a certain intersection of at least k sensors. Thus, the location of the event can be more precisely defined. This would support more fine-grained location-based services.

To determine which areas are insufficiently covered, we assume that there is a central controller in the sensor network. The central controller can broadcast the desired value of k to all sensors. Each sensor can then communicate with its neighboring sensors and then determine which segments of its perimeter are less than k-perimeter-covered. The results (i.e., insufficiently covered segments) are then sent back to the central controller. By putting all segments together, the central controller can precisely determine which areas are less than k-covered. Note that since Theorem 1 provides a necessary and sufficient condition to determine if an area in the network is k-covered, false detection would not happen.

Further actions can then be taken if certain areas are insufficiently covered. For example, the central controller can dispatch more sensors to these regions. An optimization problem is: how can we patch these insufficiently covered areas with the least number of extra sensors. This is still an open question and deserves further investigation.

3.4.2 Power Saving in Sensor Networks

Contrary to the insufficient coverage issue, a sensor network may be overly covered by too many sensors in certain areas. For example, as suggested in [32], if there are more sensors than necessary, we may turn off some redundant nodes to save energy. These sensors may be turned on later when other sensors run out of energy. Reference [32] proposes a node-scheduling scheme to guarantee that the level of coverage of the network area after turning off some redundant sensors remains the same.

Based on our result, we can solve a more general problem as follows. First, those sensor nodes who can be turned off, called *candidates*, need to be identified. A sensor s_i is a candidate if all of its neighbors are still k-perimeter-covered after s_i is removed. To do so, s_i can communicate with each of its neighbors and ask them to reevaluate their perimeter coverage by skipping s_i. If the responses from all its neighbors are positive, s_i is a candidate. After determining the candidates, each sensor can compete to enter the doze mode by running a scheduling scheme, such as that in [32], to decide how long it can go to sleep. However, [32] only considers a special case of our results with $k = 1$.

3.4.3 Hot Spots

It is possible that some areas in the network are more important than other areas and need to be covered by more sensors. Those important regions are called *hot spots*. Our solutions can be directly applied to check whether a hot spot area is k-covered or not. Given a hot spot, only those sensors whose perimeters are within or have crossings with the hot spot need to be checked. So the central controller can issue a request by identifying the hot spot. Each sensor that is within the hot spot or has crossings with the hot spot needs to reevaluate the coverage of its perimeter segment that is within the hot spot. The results in Lemma 1 and Theorem 1 are directly applicable. So a hot spot is k-covered if and only if all perimeter segments within this hot spot are k-perimeter-covered. Note that a hot spot can be defined in other shapes too.

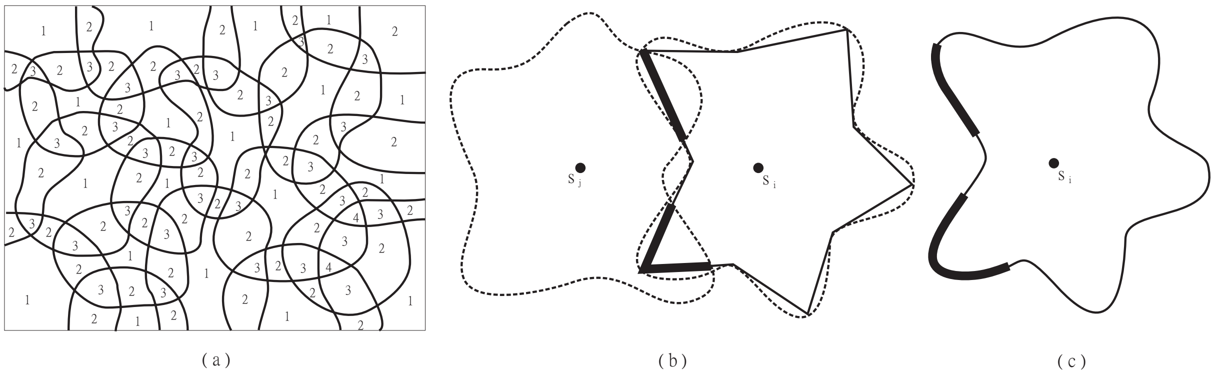


Figure 3.11: The coverage problem with irregular sensing regions: (a) coverage levels of irregular sub-regions, (b) polygon approximation of sensor s_i's sensing region, and (c) covered segments of s_i.

3.4.4 Extension to Irregular Sensing Regions

The sensing region of a sensor is not necessarily a circle. In most cases, it is location-dependent and likely irregular.[1] Fortunately, our results can be directly applied to irregular sensing regions without problem, assuming that each sensor's sensing region can be precisely defined. Observe that the sensing regions of sensors still divide the network area into sub-regions. Through Lemma 1, we can translate perimeter-covered property of sensors to area-covered property of the network. Then by Theorem 1, we can decide whether the network is k-covered. Fig. 3.11(a) shows an example.

Given two sensors' sensing regions that are irregular, it remains a problem how to determine the intersections of their perimeters. One possibility is to conduct polygon approximation. The idea is illustrated in Fig. 3.11(b), which can give the perimeter coverage in Fig. 3.11(c).

[1]The sensing region of a sensor may even be time-varying, in which case frequent reevaluation of the sensing region would be necessary. This issue is beyond the scope of this work.

Chapter 4

The Coverage Problem in a Three-Dimensional Space

In this chapter, we study the coverage problem in a three-dimensional space. We also formulate this problem as a decision problem, whose goal is to determine whether every point in the service area of the sensor network is covered by at least α sensors, where α is a given parameter and the sensing regions of sensors are modeled by balls (not necessarily of the same radius). This problem in a 2D space is solved in Chapter 3 with an efficient polynomial-time algorithm (in terms of the number of sensors). In this chapter, we show that tackling this problem in a 3D space is still feasible within polynomial time. The proposed solution can be easily translated into an efficient polynomial-time distributed protocol.

4.1 Preliminaries and Problem Statement

We are given a set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, in a three-dimensional cuboid sensing field A. Each sensor $s_i, i = 1 \ldots n$, is located at coordinate (x_i, y_i, z_i) inside A and has a sensing range of r_i. So each sensor s_i's sensing area is a *ball* centered at (x_i, y_i, z_i) with radius r_i, denoted as $B_i = (x_i, y_i, z_i, r_i)$. The *sphere* of B_i is the surface of B_i, denoted as S_i

Consider two sensors s_i and s_j which have non-empty intersecting sensing regions. The *spherical cap* $Cap(i, j)$ is the intersection of sphere S_i and ball B_j. The *circle* $Cir(i, j)$ is the intersection of spheres S_i and S_j. The *center of spherical cap* $Cap(i, j)$, denoted by $Cen(i, j)$, is the intersection of line $\overleftrightarrow{s_i s_j}$ and spherical cap $Cap(i, j)$. Given any two points p and p' on S_i, the *$\underline{g}$eodesic $\underline{d}$istance between p and p'*, denoted by $GD(p, p')$, is the minimum great circle distance

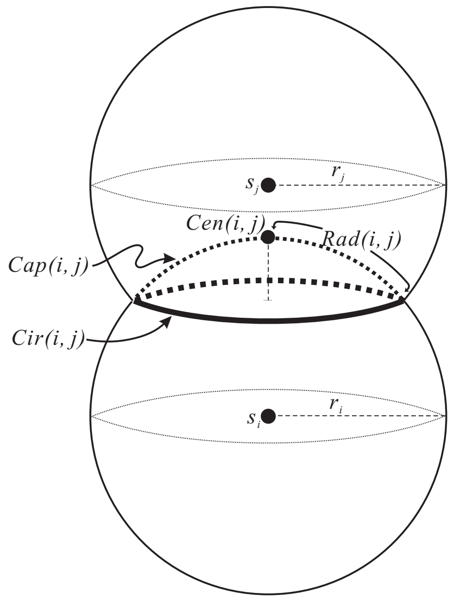


Figure 4.1: Illustration of terminologies.

between p and p' on S_i. The *radius of* $Cap(i,j)$, denoted by $Rad(i,j)$, is $GD(Cen(i,j),p)$, where p is any point on $Cir(i,j)$. Examples of these terms are illustrated in Fig. 4.1.

Definition 7 Given a natural number α, the *α-Ball-Coverage (α-BC) Problem* is a decision problem whose goal is to determine whether all points in A are α-covered or not.

4.2 The Proposed Solution

In the section, we propose an algorithm to solve the α-BC problem with time complexity $O(nd^2 \log d)$, where d is the maximum number of sensors whose sensing ranges may intersect a sensor's sensing range. Our approach does not try to look at how each point (or subspace) in A is covered by sensors because determining how A is divided by n spheres is too much complicated. Instead, our algorithm tries to determine whether the sphere of a sensor under consideration is sufficiently covered. Further, to determined whether each sensor's sphere is sufficiently covered, we look at how the circle of each spherical cap of a sensor intersected by its neighboring sensors is covered. By collecting this information from all sensors, a correct answer can be obtained. Intuitively, we reduce the decision problem from a 3D space to one in a 2D space, and then to one in a 1D space.

4.2.1 Theoretical Fundamentals

Observe that the sensing field A is divided into a number of subspaces by sensors' spheres. Each subspace's surface consists of a number of spherical segments. Because of the continuity nature, the level of coverage of a subspace can actually be derived from those of its spherical segments. Furthermore, each spherical segment must be bounded by a number of circle segments on some spherical caps. By the continuity nature again, the level of coverage of a spherical segment can actually be derived from those of its circle segments that bound the spherical segment. This is how we reduce the problem from a 3D space to a 2D space, and then to a 1D space. In the following discussion, we will use "subspace", "spherical segment", and "circle segment" to facilitate our presentation.

Definition 8 Consider any two sensors s_i and s_j. A point on sphere S_i is *sphere-covered by* s_j if it is on or within sphere S_j. We say that s_i is α*-sphere-covered* if all points on sphere S_i are sphere-covered by at least α other sensors.

Lemma 2 *If a sphere S_i is α-sphere-covered, then each subspace that is adjacent to S_i is at least α-covered.*

Proof. Since sphere S_i is α-sphere-covered, by definition each subspace that is adjacent to S_i but outside S_i is also α-covered. The subspaces inside S_i are at least $(\alpha + 1)$-covered because they are further covered by s_i[1]. □

Theorem 2 *If each sphere is α-sphere-covered, then the sensing field A is α-covered.*

Proof. Observe that each subspace in A must be bounded by some spherical segments. Since each sphere is α-sphere-covered, by Lemma 2 all subspaces are at least α-covered, which proves this theorem. □

Below, to facilitate our presentation, we translate sphere coverage into cap coverage. This allows us to look at a single sphere when examining coverage.

Definition 9 Consider any sensor s_i and its neighboring sensor s_j. A point p on S_i is *cap-covered by* $Cap(i, j)$ if p is on $Cap(i, j)$. Point p is α-cap-covered if it is cap-covered by at least α caps on S_i.

[1]In most cases, the subspaces inside S_i are $(\alpha + 1)$-covered. However, in the special case that there are k other sensors colocating with s_i and having the same sensing radiuses with s_i, these subspaces will be $(\alpha + k + 1)$-covered.

Corollary 1 *Consider any sensor s_i. If each point on S_i is α-cap-covered, then sphere S_i is α-sphere-covered.*

Proof. This corollary can be easily proved by observing the equivalence between the definitions of sphere coverage and cap coverage. □

Definition 10 Consider any sensor s_i and two of its neighboring sensors s_j and s_k. We say that a point p on $Cir(i,j)$ is *circle-covered* by $Cap(i,k)$ if p is cap-covered by $Cap(i,k)$. We say that the spherical circle $Cir(i,j)$ is α*-circle-covered* if every point on $Cir(i,j)$ is circle-covered by at least α caps on S_i other than $Cap(i,j)$.

Lemma 3 *Consider any sensor s_i and its neighboring sensor s_j. If circle $Cir(i,j)$ is α-circle-covered, then each spherical segment on S_i that is adjacent to $Cir(i,j)$ is at least α-cap-covered.*

Proof. Since circle $Cir(i,j)$ is α-circle-covered, each spherical segment on S_i that is adjacent to $Cir(i,j)$ but outside $Cap(i,j)$ is also α-cap-covered. The spherical segments on S_i inside $Cap(i,j)$ are at least $(\alpha+1)$-cap-covered because they are further covered by $Cap(i,j)$[2]. □

Theorem 3 *Consider any sensor s_i and each of its neighboring sensors s_j. If each circle $Cir(i,j)$ is α-circle-covered, then the sphere S_i is α-cap-covered.*

Proof. Observe that each spherical segment on S_i must be bounded by some circle segments. Since each circle is α-cap-covered, by Lemma 3 all spherical segments on S_i are at least α-cap-covered, which proves this theorem. □

4.2.2 Determining the Intersection of Spherical Caps

The above derivation implies that to determine how A is covered, it is sufficient to determine how each circle is covered. To determine circle coverage, consider any two spherical caps $Cap(i,j)$ and $Cap(i,k)$ on sphere S_i of a sensor s_i. There are two cases:

[2] In most cases, these spherical segments are $(\alpha+1)$-cap-covered. However, in the special case that there are k other caps colocating with the current $Cap(i,j)$, these spherical segments will be $(\alpha+k+1)$-cap-covered. Note that colocating caps may appear when two spheres intersect with another sphere on the same circle.

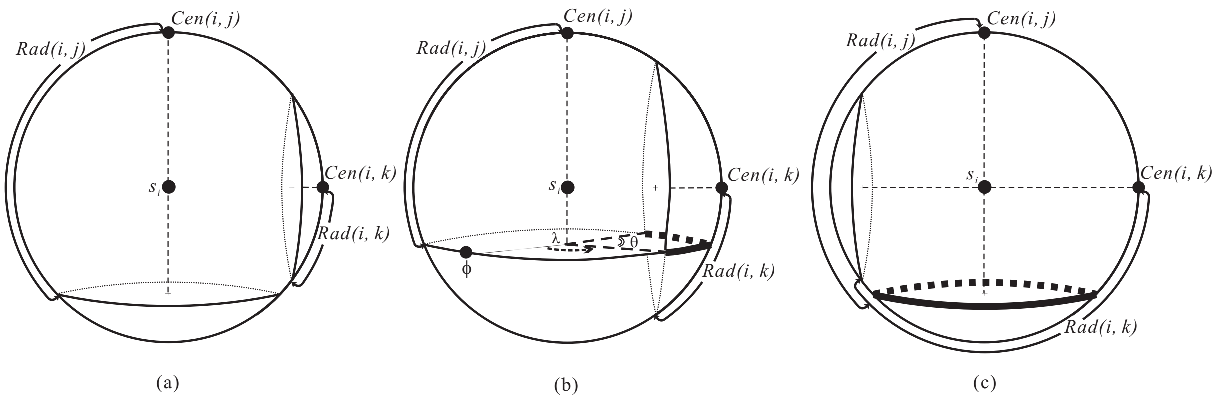


Figure 4.2: The relationship between $Cap(i, j)$ and $Cap(i, k)$: case 1.

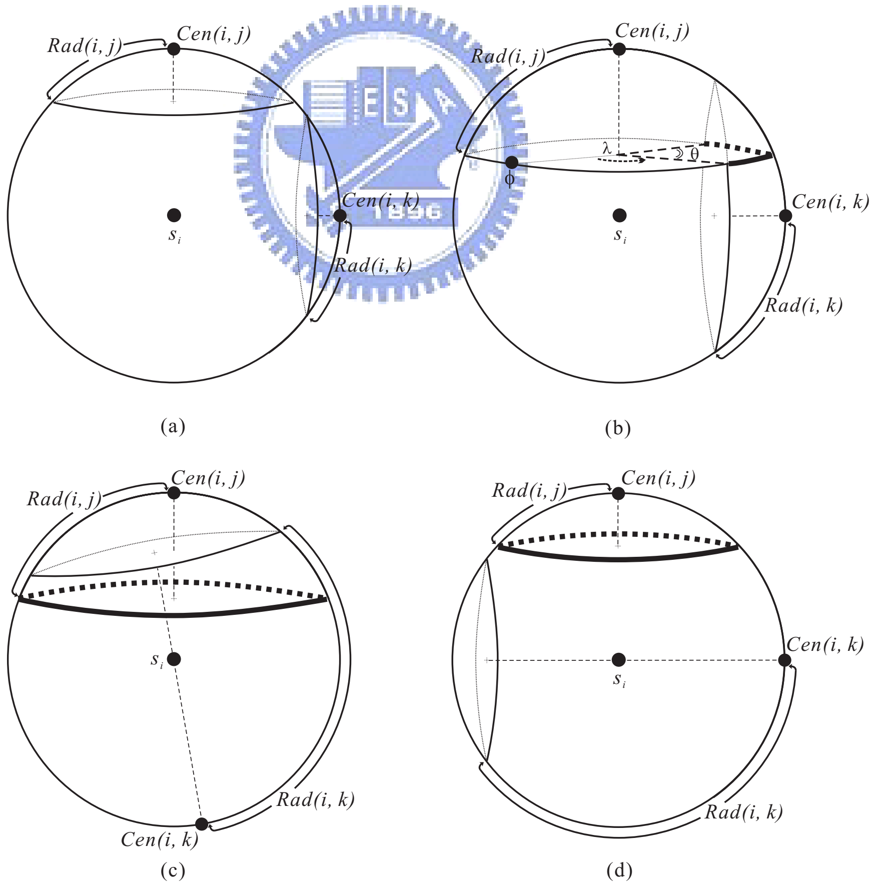


Figure 4.3: The relationship between $Cap(i, j)$ and $Cap(i, k)$: case 2.

1: The center of $Cap(i,k)$, $Cen(i,k)$, is inside $Cap(i,j)$, i.e., $GD(Cen(i,j), Cen(i,k)) \leq Rad(i,j)$.

   (i) If $Rad(i,k) < Rad(i,j) - GD(Cen(i,j), Cen(i,k))$, then $Cap(i,j)$ is not circle-covered by $Cap(i,k)$ (refer to Fig. 4.2(a)).

   (ii) If $Rad(i,j) - GD(Cen(i,j), Cen(i,k)) \leq Rad(i,k) \leq GD(Cen(i,j), Cen(i,k)) + Rad(i,j)$, then the arch of $Cir(i,j)$ falling in the angle $[\lambda, \lambda + \theta]$ is circle-covered by $Cap(i,k)$ (refer to Fig. 4.2(b)).

   (iii) If $Rad(i,k) > Rad(i,j) + GD(Cen(i,j), Cen(i,k))$, then the whole range $[0, 2\pi]$ of $Cap(i,j)$ is circle-covered by $Cap(i,k)$ (refer to Fig. 4.2(c)).

2: The center of $Cap(i,k)$, $Cen(i,k)$, is outside $Cap(i,j)$, i.e., $GD(Cen(i,j), Cen(i,k)) > Rad(i,j)$.

   (i) If $Rad(i,k) < GD(Cen(i,j), Cen(i,k)) - Rad(i,j)$, then $Cap(i,j)$ is not circle-covered by $Cap(i,k)$ (refer to Fig. 4.3(a)).

   (ii) If $GD(Cen(i,j), Cen(i,k)) - Rad(i,j) \leq Rad(i,k) \leq GD(Cen(i,j), Cen(i,k)) + Rad(i,j)$, then the arch of $Cir(i,j)$ falling in the angle $[\lambda, \lambda + \theta]$ is circle-covered by $Cap(i,k)$ (refer to Fig. 4.3(b)). Note that it is possible that there is no intersection between $Cir(i,j)$ and $Cir(i,k)$, but $Cir(i,j)$ is fully covered by $Cap(i,k)$, as illustrated in Fig. 4.3(c).

   (iii) If $Rad(i,k) > GD(Cen(i,j), Cen(i,k)) + Rad(i,j)$, then the whole range $[0, 2\pi]$ of $Cir(i,j)$ is circle-covered by $Cap(i,k)$ (refer to Fig. 4.3(d)).

4.2.3 The Complete Algorithm

Below, we propose an $O(d^2 \log d)$ algorithm to determine whether a sensor is α-sphere-covered or not. The algorithm can be executed either in a centralized or in a fully distributed manner independently by each sensor. First, each sensor has to collect how its neighboring sensors intersect with itself and calculate the corresponding spherical caps. Next, it has to figure out the relationship between spherical caps, as described above. Then we can determine the level of circle coverage of each circle. After each cap's circle coverage level is determined, the sensor's sphere coverage level can be found out, which in turn gives the overall coverage of A. The detail algorithm to be run by each sensor s_i is listed below.

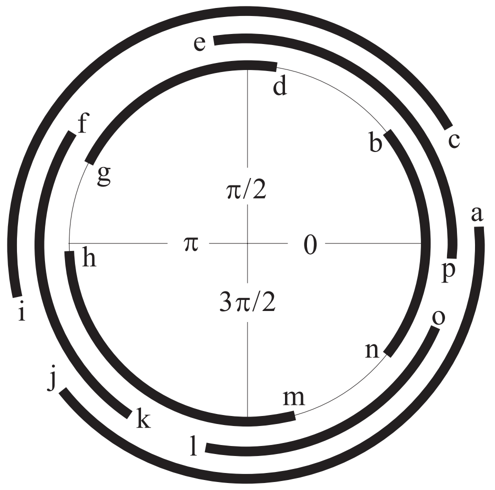


Figure 4.4: An example to determine the coverage of a circle.

1) For each neighboring sensor s_j of s_i, do the following.

   a) Calculate the circle $Cir(i,j)$ of $Cap(i,j)$.

   b) For each neighbor $s_k \neq s_j$ of s_i, we determine how $Cap(i,k)$ intersects with $Cir(i,j)$. Specifically, we calculate the angle of $Cir(i,j)$ that is circle-covered by $Cap(i,k)$, denoted by $[\theta^j_{k,L}, \theta^j_{k,R}]$.

   c) For all angles $[\theta^j_{k,L}, \theta^j_{k,R}]$ found in step b), place points $\theta^j_{k,L}$ and $\theta^j_{k,R}$ on a line segment $[0, 2\pi]$, and then sort all these points in an ascending order into a list L_j.

   d) (sketched) Traverse the line segment $[0, 2\pi]$ by sequentially visiting each point in the sorted list L_j to determine the circle coverage of $Cir(i,j)$, denoted by cc_j.

   end for.

2) The sphere coverage of s_i is the minimum circle coverage of all circles on S_i, i.e., $min_{\text{neighbor } s_j}\{cc_j\}$.

Let d be the maximum number of sensors neighboring to a sensor ($d \leq n$). Step 1a, 1b, 1c, and 1d have time complexities of $O(1), O(d), O(d \log d)$, and $O(d)$, respectively. So the complexity of step 1 is $O(d^2 \log d)$, which is also the complexity of the whole algorithm for one sensor.

The step 1d, though sketched, can be easily implemented as follows. Whenever an element $\theta^j_{k,L}$ is traversed, the level of coverage should be increased by one. Whenever an element $\theta^j_{k,R}$ is traversed, the level of coverage should be decreased by one. An example is shown in Fig. 4.4.

The point on angle 0 can be easily determined to be 3. When visiting points c, d, f, h, j, l, n, and p (resp., points a, b, e, g, i, k, m, and o), the level of coverage should be increased (resp., decreased) by 1.

Below, we comment on several special cases, which we leave not addressed on purpose for simplicity in the above discussion. First, it is possible that a sensor's sensing range is fully covered by another sensor's, i.e., a sensing ball is entirely inside another sensing ball. These two spheres do not have any intersection. Alternatively, we can regard the whole sphere of the smaller one as a special spherical cap. So we can simply increase the sphere coverage level of the smaller sphere by one after executing our algorithm. Another boundary case is that some sensors' sensing ranges may exceed the sensing field A. If so, we can simply assign the spherical segments falling outside A as ∞-sphere-covered.

Chapter 5

Ensuring Both Coverage and Connectivity

For a sensor network to operate successfully, sensors must maintain both sensing coverage and network connectivity. This issue has been studied in [35, 43], both of which reach a similar conclusion that coverage can imply connectivity as long as sensors' communication ranges are no less than twice their sensing ranges. In this chapter, we investigate this issue from a different angle and propose more general decentralized solutions that do not rely on the above assumption. Hence, the results in [35, 43] can be regarded as special cases of what proposed in this chapter.

5.1 Preliminaries and Problem Statement

We are given a set of sensors, $S = \{s_1, s_2, \ldots, s_n\}$, in a two-dimensional area A. Each sensor s_i, $i = 1 \ldots n$, is located at a known coordinate (x_i, y_i) inside A and has a sensing distance of r_i and a communication distance of c_i. For simplicity, we assume there are no two sensors located in the same location. So, s_i can detect an object/event located within a distance of r_i from itself and talk to another sensor within a distance of c_i. Note that we make no assumption about the relationship of r_i and c_i. However, unidirectional links are excluded, so packets can only be sent on bidirectional links.

Definition 11 The sensor network is said to be *1-connected* if there is at least one path between any two sensors. The sensor network is said to be *k-connected* if there are at least k disjointed paths between any two sensors.

We formulate the general form of coverage and connectivity problem as follows.

Definition 12 Given any two integers k_1 and k_2, the k_1*-Covered and* k_2*-Connected Problem*, or the (k_1, k_2)*-CC problem*, is a decision problem whose goal is to determine whether the sensor network is k_1-covered and k_2-connected.

5.2 The Proposed Solutions

In this section, we propose theoretical foundations and a distributed protocol to solve the (k_1, k_2)-CC problem. We make no assumption on the relationship between communication distances and sensing distances. Following the model in Chapter 3 and Chapter 4, our approach also looks at how the perimeter of each sensor's sensing range is covered by its neighbors, and whether there is a link/path to each of its neighbors. We show conditions for a sensor network to be k-covered and k-connected, and to be k-covered and 1-connected. We also show under what conditions a sensor network may provide sufficient coverage by multiple connected components.

5.2.1 Theoretical Fundamentals

The definition of perimeter coverage has been proved useful to determine the coverage level of a sensor network in Chapter 3. Below, we define similar notations based on stronger conditions.

Definition 13 Consider any sensor s_i. The *neighboring set* of s_i, denoted as $N(i)$, is the set of sensors each of whose sensing region intersects with s_i's sensing region.

Definition 14 Consider any sensor s_i. We say that s_i is k*-direct-neighbor-perimeter-covered*, or k*-DPC*, if s_i is k-perimeter-covered and s_i has a link to each node in $N(i)$. Similarly, we say that s_i is k*-multihop-neighbor-perimeter-covered*, or k*-MPC*, if s_i is k-perimeter-covered and s_i has a (single- or multi-hop) path to each node in $N(i)$.

Lemma 4 *Consider any two sensors s_i and s_j. If each sensor in S is 1-MPC, there must exist a communication path between s_i and s_j.*

Proof. This proof is by construction. If s_i's sensing region intersects with s_j, by Definition 14, there must exist a path between s_i and s_j, which proves this lemma. Otherwise, draw a line segment L connecting s_i and s_j, as illustrated in Fig. 5.1(a). Let L intersect s_i's perimeter at

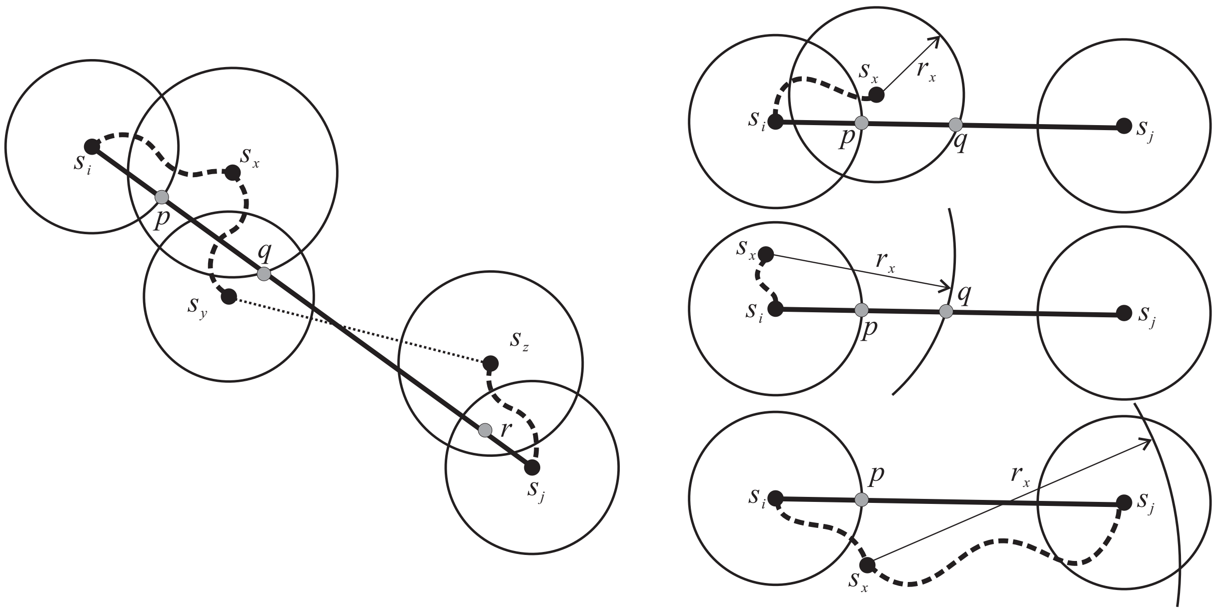


Figure 5.1: Proof of Lemma 4: (a) the path construction, and (b) possible cases of s_x.

point p. Since s_i is 1-MPC, by Definition 14, there must exist a sensor s_x in $N(i)$ which covers p and has a path to s_i. In addition, either s_x must cover s_j, or s_x's perimeter must intersect L at a point, namely q, which is closer to s_j than p is. Fig. 5.1(b) shows several possible combinations of s_x and r_x. In the former case, by Definition 14, there must exist a path between s_x and s_j, and thus s_i and s_j, which proves this lemma. In the later case, there must exist another sensor s_y in $N(x)$ which covers q. We can repeat the above argument until a sensor s_z is found which either covers s_j or intersect L at a point, say r, inside s_j's sensing range. In either case, there must exist a path between s_z and s_j, which proves this lemma. □

Theorem 4 *A sensor network is k-covered and 1-connected iff each sensor is k-MPC.*

Proof. For the "if" part, we have to guarantee both the coverage and connectivity. The fact that the network is k-covered has been proved by Theorem 1 because each sensor which is k-MPC is also k-perimeter-covered. In addition, Lemma 4 can guarantee that the network is 1-connected, hence proving the "if" part.

For the "only if" part, we have to show that each sensor is k-perimeter-covered and has a path to each sensor whose sensing region intersects with its region. The first concern can be ensured by Theorem 1, while the second concern can be ensured by the fact that the network is 1-connected. □

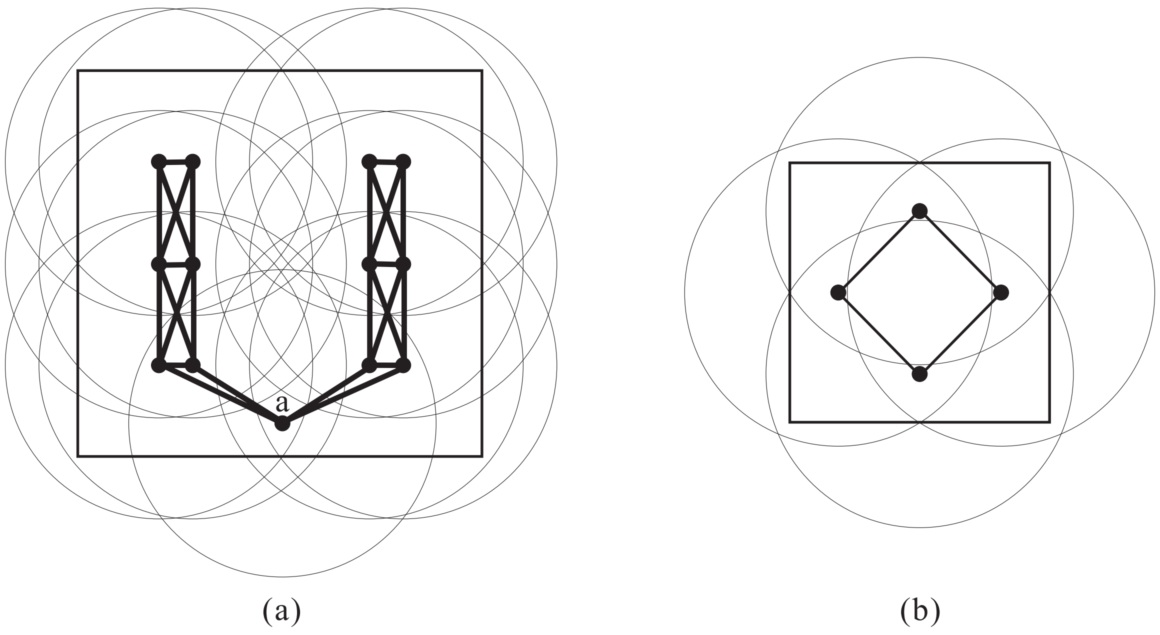


Figure 5.2: Observations of Theorem 4 and Theorem 5: (a) The network is 2-covered and 1-connected. The removal of sensor a will disconnect the network, and (b) The network is 2-covered and 2-connected but no sensor is 2-DPC. Note that the sensing and communication ranges of each sensor are the same and are represented by circles.

Theorem 5 *A sensor network is k-covered and k-connected if each sensor is k-DPC.*

Proof. Coverage has been guaranteed by Theorem 1 since a sensor which is k-DPC is k-perimeter-covered by definition. For the connectivity part, if we remove any $k-1$ nodes from the network, it is not hard to see that each of the rest of sensors must remain 1-DPC. This implies that these sensors are also 1-MPC, and by Lemma 4 there must exist a path between any pair of these sensors. As a result, the network is still connected after the removal of any $k-1$ nodes, which proves this theorem. □

Below we make some observations about Theorem 4 and Theorem 5. First, a major difference is that Theorem 4 can guarantee only 1 connectivity, while Theorem 5 can guarantee k connectivity. This is because, in a network where each sensor is k-MPC, the removal of any sensor may disconnect the network. For example, in the network in Fig. 5.2(a), each sensor is 2-MPC. By Theorem 4, the network is 2-covered and 1-connected. However, if we remove sensor a, the network will be partitioned into two components. Interestingly, although the network remains 2-covered, it is no longer connected.

Second, the reverse direction of Theorem 5 may not be true. That is, if a network is k-covered and k-connected, sensors in this network may not be k-DPC. Fig. 5.2(b) shows an example.

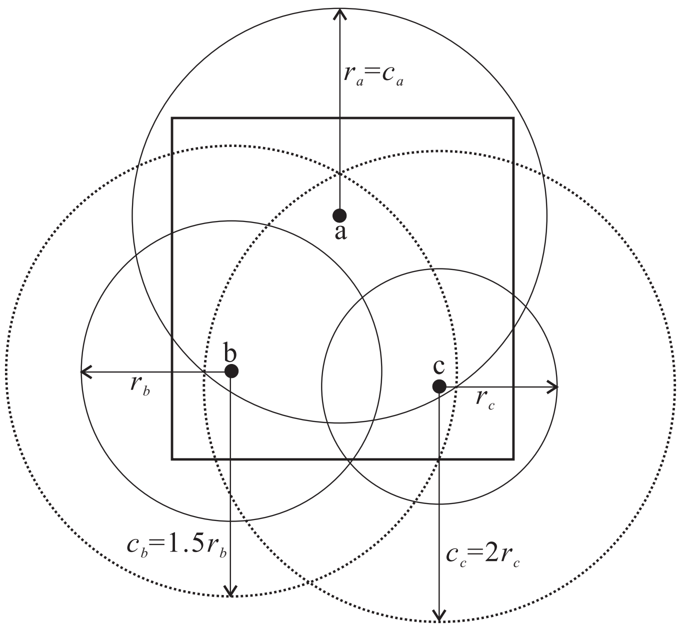


Figure 5.3: An example to compare Theorem 5 with results in [35, 43]. Solid circles and dotted circles are sensors' sensing ranges and communications ranges, respectively.

example in which the network is 2-covered and 2-connected. However, each node has a neighbor (with overlapping sensing range) to which there is no direct communication link.

Third, Theorem 5 is stronger than the results in [35, 43]. It is clear that when two sensors have overlapping sensing range, there is a direct communication link between these two sensors if the communication distance is at least twice the sensing distance. So what can be determined by [35, 43] can also be determined by Theorem 5. Furthermore, when the above assumption does not exist, Theorem 5 may still work while [35, 43] do not. For example, Theorem 5 can determine that the network in Fig. 5.3 is 1-covered and 1-connected, when some sensors' communication ranges are less than twice their sensing ranges.

5.2.2 Looser Connectivity Conditions

Definition 15 The *direct neighboring set* of s_i, denoted as $DN(i)$, is the set of sensors each of which has a communication link to s_i and whose sensing region intersects with s_i's sensing region. Similarly, the *multi-hop neighboring set* of s_i, denoted as $MN(i)$, is the set of sensors each of which has a (single- or multi-hop) path to s_i and whose sensing region interests with s_i's.

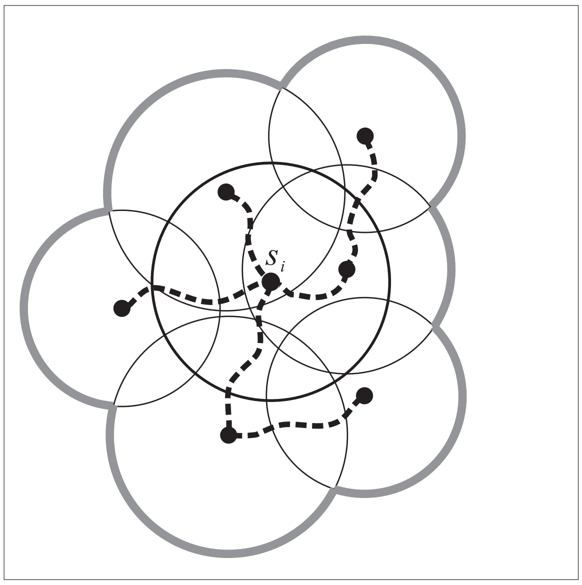


Figure 5.4: Proof of the Lemma 5.

Definition 16 Consider any sensor s_i. We say that s_i is *k-loose-direct-neighbor-perimeter-covered*, or *k-LDPC*, if s_i is k-perimeter-covered by and only by nodes in $DN(i)$. Similarly, we say that s_i is *k-loose-multihop-neighbor-perimeter-covered*, or *k-LMPC*, if s_i is k-perimeter-covered by and only by nodes in $MN(i)$.

We comment that for any sensor s_i, $DN(i) \subseteq MN(i) \subseteq N(i)$. So the definition that s_i is k-LDPC is looser than that s_i is k-DPC in the sense that k-DPC guarantees that there is a link from s_i to each of $N(i)$, but k-LDPC only guarantees that there is a link from s_i to each of $DN(i)$. The definition of k-LMPC is looser than that of k-MPC in a similar sense.

Lemma 5 *If each sensor in S is 1-LMPC, then the network can be decomposed into a number of connected components each of which 1-covers the sensing field A.*

Proof. This proof is by construction. For any sensor s_i, we try to construct a connected component which fully covers A. (However, the proof does not guarantee that s_i has a path to every sensor.) If s_i's sensing region can fully cover A, the construction is completed. Otherwise, by Definition 16, nodes in $MN(i)$ must perimeter-cover s_i's perimeter and each has a path to s_i, as illustrated in Fig. 5.4. In addition, nodes in $MN(i)$ together with s_i form a larger coverage region which is bounded by perimeters of nodes in $MN(i)$. If A is already fully covered by this region, the construction is completed. Otherwise, since each sensor is 1-LMPC, we can repeat similar arguments by extending the coverage region, until the whole field A is covered. $\square$

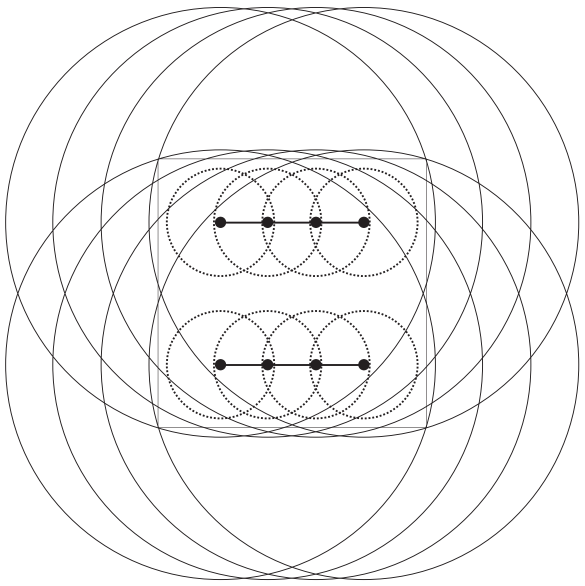


Figure 5.5: An example of two connected components each of which 1-covers A.

Theorem 6 *A sensor network can be decomposed into a number of connected components each of which k-covers A iff each sensor is k-LMPC.*

Theorem 7 *A sensor network can be decomposed into a number of k-connected components each of which k-covers A if each sensor is k-LDPC.*

The proof of Theorem 6 (respectively, Theorem 7) is similar to Theorem 4 (respectively, Theorem 5) by replacing Lemma 4 with Lemma 5. An example of Theorem 6 is shown in Fig. 5.5. Due to relatively small communication ranges compared to sensing ranges, the network is partitioned into two connected components. However, each component still provides sufficient 1-coverage.

To summarize, Theorem 6 and Theorem 7 only guarantee that the network can be sufficiently covered by each connected component, while Theorem 4 and Theorem 5 can guarantee both coverage and connectivity of the whole network. When $DN(i) = N(i)$ or $MN(i) = N(i)$ for each sensor s_i, these theorems converge. Also observe that Theorem 6 and Theorem 7 are more practical because each sensor only needs to collect its reachable neighbors' information to make its decision. Most applications can be satisfied if a subset of sensors is connected and can provide sufficient coverage. The redundancy caused by multiple components may be eliminated by a higher level coordinator, such as the base station, to properly schedule each component's working time such that no two components of the network are active at the same time.

5.2.3 Protocols to Determine Coverage and Connectivity

The above results imply that to determine how a sensor network is covered and connected, it is sufficient to determine how each sensor's perimeter is covered by its neighbors. The decision procedure can be executed in a fully distributed manner independently by each sensor.

For a sensor to determine how its perimeter is covered, first it has to collect how its one-hop neighboring sensors' sensing regions intersect with its and calculate the level of perimeter coverage. Periodical *BEACON* messages can be sent to carry sensors' location and sensing range information. After receiving such BEACON messages, a sensor can determine who are its direct neighbors and how its perimeter is covered by them. The detail algorithm to determine a sensor's perimeter coverage can be found in Chapter 3. If the level of perimeter coverage is at least k in this step, we can determine that this sensor is k-LDPC.

If a sensor's perimeter is not sufficiently covered by its one-hop neighbors, a *QUERY* message is flooded to find out more sensors whose sensing regions interest with its region. The flooding can be a localized flooding by limiting its range within some hop count. Each sensor who receives the QUERY message has to check if its sensing region intersects with the source node's. If so, a *REPLY* message is sent to the source node. After a predefined timeout period, the source can calculate its level of perimeter coverage based on received REPLY messages. If the level of perimeter coverage is at least k in this step, we can determine that this sensor is k-LMPC. Otherwise, we can take an incremental approach by flooding another QUERY with a larger hop limit, until the desired k is reached or the whole network is flooded.

After the above steps, each sensor can report its exploring result to the base station or a centralized sensor (note that aggregation mechanisms may be used in the reporting, but we omit the details). Then the base station can determine the coverage and connectivity of the network. There are three possible cases. If each sensor is k-LDPC, the network is k-covered and k-connected. If some sensors are k-LMPC while others are k-LDPC, the network is k-covered and 1-connected. If there exist sensors that are neither k-LDPC nor k-LMPC, the result is undecided. In this case, it is possible that the network is still sufficiently covered but is partitioned. For example, if we remove sensor a in Fig. 5.2(a), the network is 2-covered, but the proposed theorems can not detect such case.

Chapter 6

Decentralized Energy-Conserving and Coverage-Preserving Protocols

Since sensors are usually powered by batteries, sensors' on-duty time should be properly scheduled to conserve energy. If some nodes share the common sensing region and task, then we can turn off some of them to conserve energy and thus extend the lifetime of the network. In this chapter, we propose decentralized protocols that schedule sensors' active and sleeping periods to prolong the network lifetime while maintain the sensing field sufficiently covered. The proposed protocols are similar to the model in [40]. However, our approach can significantly reduce the computational complexity incurred on each sensor. In addition, our approach can relieve the inaccuracy caused by using gird points to calculate each sensor's working schedules in [40].

6.1 Preliminaries and Problem Definition

We are given a set of sensors, $S = \{s_1, s_2, \dots, s_n\}$, in a two-dimensional area A. Each sensor s_i, $i = 1, \dots, n$, knows its own location (x_i, y_i) inside A and has a sensing range of r_i, i.e., it can monitor any point that is within a distance of r_i from it. Each sensor is able to switch between the *active* mode and the *sleeping* mode. While active, a sensor can conduct sensing tasks and communicate with neighbors. While sleeping, a sensor turns off both its sensing and communication devices to conserve energy. In addition, each sensor s_i is aware of its own remaining energy, denoted as E_i, all the time.

Definition 17 Given a threshold value γ, $0 < \gamma \leq 1$, the *lifetime(γ)* of a sensor network is the duration from the network being started until the first moment when the ratio of area over A that is covered is below the threshold γ.

For example, $lifetime(1)$ is the duration until the first location in A is no longer covered. Our goal is to develop an energy-efficient coverage-preserving protocol for the wireless sensor network by scheduling sensors' active and sleeping periods such that the lifetime of the network is as long as possible.

6.2 A Basic Coverage-Preserving Protocol

In this section, we first give an overview of the our protocol. Further parameter-setting criteria will be discussed afterwards, followed by the complexity analysis.

6.2.1 Protocol Structure

The proposed protocol is similar to the model in [40]. However, our approach can significantly reduce the computational complexity incurred on each sensor. The time axis of each node is divided into a sequence of *working cycles*, each of the same length T_{w_cycle}. The working cycles of sensors are assumed to be roughly synchronous. (As will be seen later, global time synchronization is unnecessary in our protocol.) Each working cycle consists of two phases, an *initialization phase* of length T_{init} and a *sensing phase* of length T_{sen}. The initialization phase is for sensors to exchange information and use the information to calculate their working schedules for energy conservation purpose. Then in the sensing phase sensors will switch between active and sleeping modes according to their working schedules.

Fig. 6.1 illustrates one working cycle. During the initialization phase, each s_i has to wake up and broadcast a *HELLO* packet containing the following information: (x_i, y_i), r_i, and Ref_i, where Ref_i is generated from some random process. Based on the HELLO packets received from neighbors, s_i can calculate it own working schedule in the sensing phase. Note that to avoid possible collisions, a random backoff should be taken before HELLO. The sensing phase is divided into r rounds with equal duration T_{rnd}, i.e., $T_{sen} = r \times T_{rnd}$. In each round, the active period of s_i is from $Ref_i - Front_i$ until $Ref_i + End_i$ after the round begins. The details to generate Ref_i, $Front_i$, and End_i will be addressed later. Note that after each working cycle, Ref_i should be regenerated so as to fairly distribute energy consumption among sensors. Also

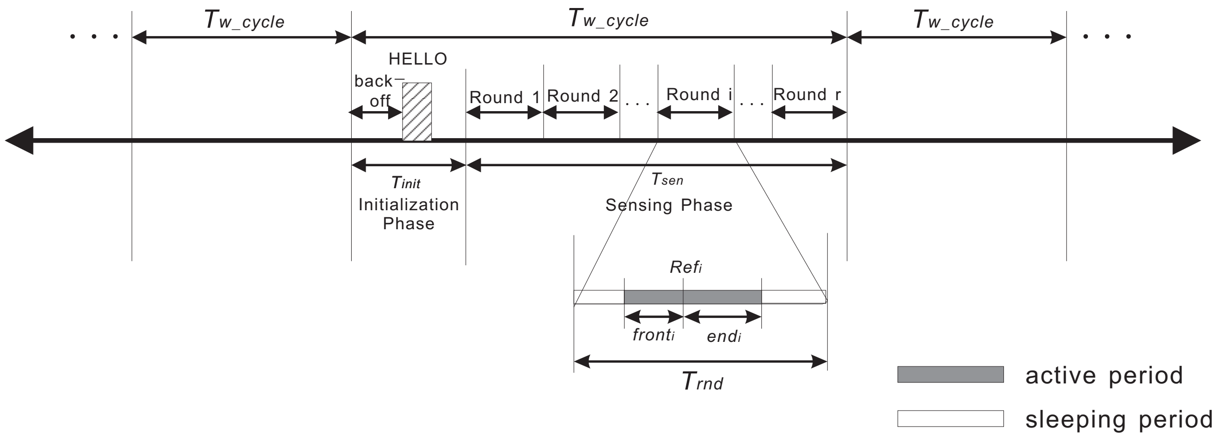


Figure 6.1: One working schedule of a sensor node.

note that the above schedule only reflects the behavior of sensors when only monitoring job is needed and no action is taken by the network. When actions need to be taken (such as events being detected), sensors may wake up each other, but this is beyond the scope of this paper.

Next, we present a basic method for sensor s_x to select Ref_x, $Front_x$, and End_x. The method is a modification of what discussed in [40]. First, s_x generates a reference time Ref_x which is uniformly distributed in $[0, T_{rnd})$. Then, from HELLO packets received, s_x should maintain a neighbor table which contains all its neighbors' locations, sensing ranges, and reference times. The parameters $Front_x$ and End_x should be carefully calculated to ensure that the sensing area is sufficiently covered. To achieve this goal, we utilize a theorem which is first stated in [10]. This theorem claims that, if all intersection points between any two sensors' sensing ranges and between any sensor's sensing range and the boundary of A are sufficiently covered, the target area is sufficiently covered. This result is also used in [35] and [43] to guarantee the coverage of a sensor network. More specifically, for each intersection point, we have to schedule at least one sensor to be on-duty at any moment among all sensors which cover the point. This leads to an efficient distributed protocol.

Consider any sensor s_x. Let the set of intersection points inside s_x's sensing range be P. For each point $p \in P$, s_x has to calculate a $Front_x^p$ and a End_x^p as follows. First, from s_x's neighbor table, s_x can find all sensors that also covers point p. Then s_x sorts these sensors (including itself) into a list L_p in ascending order of their reference times. We then define:

$$Front_x^p = [(Ref_x - prev(Ref_x)) \bmod T_{rnd}]/2 \tag{6.1}$$

$$End_x^p = [(next(Ref_x) - Ref_x) \bmod T_{rnd}]/2 \tag{6.2}$$

where $prev(Ref_x)$ and $next(Ref_x)$ are the reference times before and after Ref_x in the list L_p, respectively. Note that here we consider L_p as a circular list, i.e., the one next to the last item in L_p is the first item in L_p, and vice versa. For ease of presentation, the time period in a round T_{rnd} is also treated in a circular manner. For example, when a negative time t is referred, we actually mean t mod T_{rnd}. Intuitively, Eq. (6.1) and Eq. (6.2) are designed to have sensors in L_p cooperatively cover point p in a time-division manner. For two consecutive reference times in L_p, the corresponding two sensors will divide their responsibility at the middle point of their reference times, such that one covers p before the middle point, and the other does after the middle point. This is formally stated in the following lemma.

Lemma 6 *If each sensor $s_x \in L_p$ is active in the duration $[Ref_x - Front_x^p, Ref_x + End_x^p]$ (in the circular sense), then p is covered by exactly one sensor in each round.*

Fig. 6.2(a) shows an example. Intersection point p is covered by sensors s_1 and s_2. Let $T_{rnd} = 20$, and the reference times of s_1, s_2, s_3, and s_4 be 2, 9, 11, and 16, respectively. So we have $Front_1^p = [(2-9) \mod 20]/2 = 6.5$, and $End_1^p = [(9-2) \mod 20]/2 = 3.5$. Similarly, $Front_2^p = 3.5$ and $End_2^p = 6.5$. The result is shown in Fig. 6.2(b). As can be seen, p is covered by exactly one sensor at any moment.

It is not hard to see that the above scheduling may result in inconsistent active times, considering the existence of multiple intersection points in a network. To ensure each intersection point is covered, the active period of a sensor should be the union of schedules obtained from all intersection points under its coverage. So we define:

$$Front_x = \max_{\forall p \in P}\{Front_x^p\} \tag{6.3}$$

$$End_x = \max_{\forall p \in P}\{End_x^p\}. \tag{6.4}$$

Theorem 8 *If each sensor s_x is active in the duration $[Ref_x - Front_x, Ref_x + End_x]$ (in the circular sense), then the whole sensor network is covered in each round.*

For example, the integrated schedule of sensor s_1 in Fig. 6.2(a) is shown in Fig. 6.2(c).

6.2.2 Energy-Based Scheduling

The above basic scheduling does not consider status of sensors – reference times are randomly selected, and sensors equally divide their responsibility to cover the sensing field. In this subsection, we try to utilize sensors' remaining energy to balance their energy consumption and

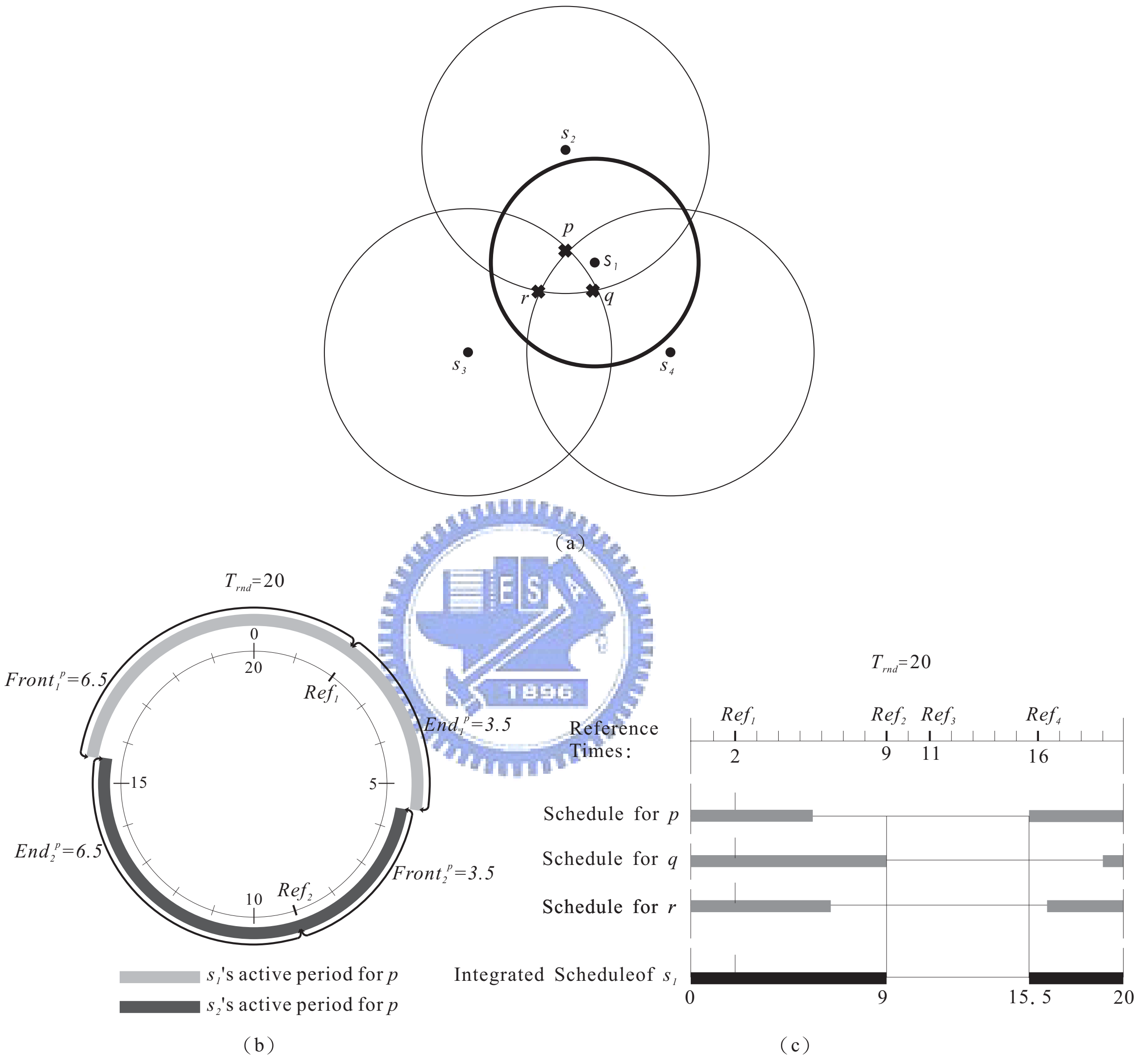


Figure 6.2: (a) a 4-sensor example, (b) schedules of s_1 and s_2 for intersection p, and (c) the integrated schedule of s_1.

prolong network lifetime. Note that this require each sensor to broadcast its remaining energy in the HELLO packet.

We first discuss how to choose reference times. For any intersection point p, the interval between two adjacent reference times in L_p will affect the corresponding sensors' on-duty times in a round. Therefore, the reference times of sensors with more energy should be placed more sparsely on the time line than those with less energy. To achieve this goal, each round is logically separated into two zones with different lengths, $[0, \frac{3T_{rnd}}{4})$ and $[\frac{3T_{rnd}}{4}, T_{rnd})$. Sensors with more energy should randomly choose their reference times from the larger zone , while sensors with less remaining energy should choose from the smaller zone. The criteria to determine sensors' remaining energy may be based on some agreement, such as a threshold. Alternatively, if a node finds its remaining energy ranked top 50% among its neighbors, it chooses from $[0, \frac{3T_{rnd}}{4})$. Otherwise, it chooses from $[\frac{3T_{rnd}}{4}, T_{rnd})$.

Parameters $Front_x$ and End_x of sensor s_x can also be tuned according to the remaining energies. For any point p, we can modify Eq. (6.1) and Eq. (6.2) according to the ratio of remaining energies of two nodes as follows:

$$Front_x^p = [(Ref_x - prev(Ref_x)) \bmod T_{rnd}] \times \frac{E_i}{E_i + E_{i-1}} \tag{6.5}$$

$$End_x^p = [(next(Ref_x) - Ref_x)) \bmod T_{rnd}] \times \frac{E_i}{E_i + E_{i+1}} \tag{6.6}$$

6.2.3 Complexity Analysis and Discussion

The computational complexity of the proposed protocol is analyzed below. To calculate its working schedule, a sensor first looks at its neighbor table and extracts reference times of its neighbors. Suppose that a node has at most d neighbors. Then sorting these reference times takes time $O(d \log d)$. The maximum number of intersection points covered by sensor is $O(d^2)$. For each intersection point, a sensor has to find out which nodes covering the point, which takes time $O(d)$. So, the calculation of working schedule for all intersection points takes time $O(d^3)$. Finally, calculating Eq. (6.3) and Eq. (6.4) takes time $O(d^2)$. Therefore, a complexity of $O(d^3)$ is incurred on each node to decide its working schedule. Note that the energy-based scheduling does not incur higher cost than the basic scheme.

Next, we compare our scheme with that in [40]. The complexity of [40] is related to the grid size of the entire region (while our protocol is independent of the grid size). Suppose that each grid has a width of g and each sensor's sensing range is r. Then there are approximately $\frac{\pi r^2}{g^2}$

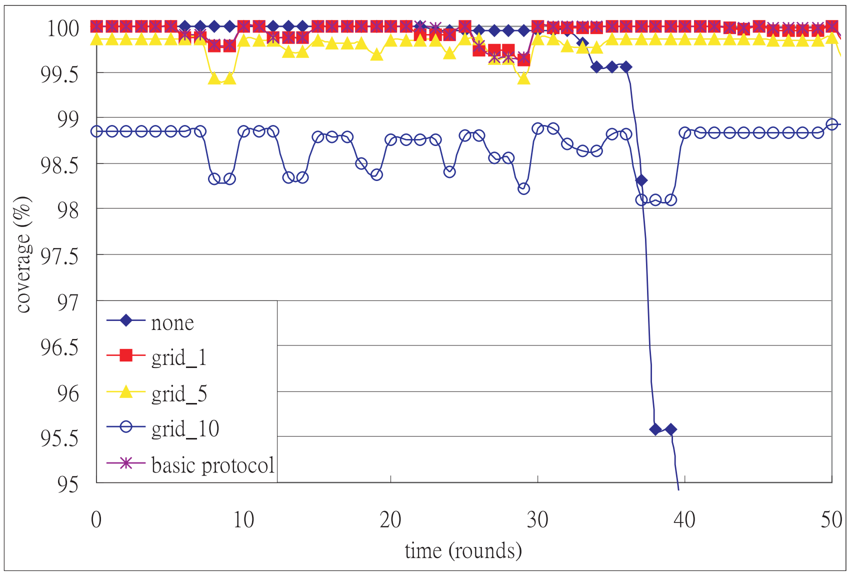


Figure 6.3: The ratio of covered area achieved by our basic scheme and the scheme in [40] with different grid size.

grids to be taken care of by a sensor. As a result, it takes time $O(d\frac{\pi r^2}{g^2})$ for a sensor to decide its working schedule. In addition, grid approximation is sometimes inaccurate.

Next, we discuss the effect of loss of HELLO messages. HELLO messages carry important information to neighboring hosts. If a sensor misses a neighbor's HELLO, it may compile an incomplete list L_p. However, the correctness of our protocol is not affected, because this only results in longer on-duty time (observe that the functions $prev()$ and $next()$ may return reference times that are farther away than they should be). As a result, the coverage is still guaranteed even in loss of HELLOs.

6.3 Simulation Results

We have developed a simulator to compare the performance of the proposed energy-conserving protocols. The simulation environment is a 100×100 square area, on which 150 sensors are randomly generated. Each sensor has a sensing range of 25. In addition, each sensor will reschedule again every 5 rounds, i.e., a cycle includes 5 rounds. The initial energy settings of sensors are uniformly distributed between $1 \sim 50\ T_{rnd}$, i.e., if a sensor keep active all the time, it will run out it energy after $1 \sim 50$ rounds.

First, we compare our scheme with the scheme in [40]. The result is shown in Fig. 6.3. We look at the ratios of covered area achieved by different schemes when time goes by. Those three

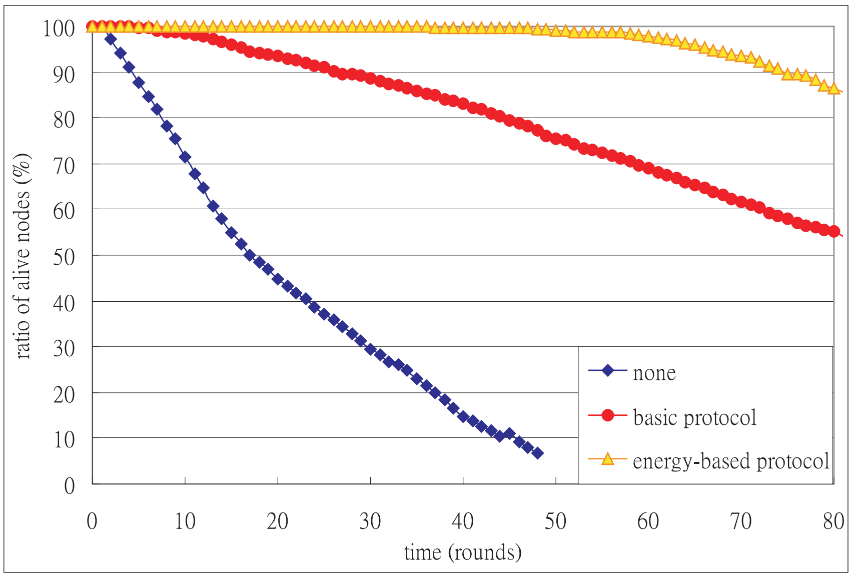


Figure 6.4: The ratio of alive nodes.

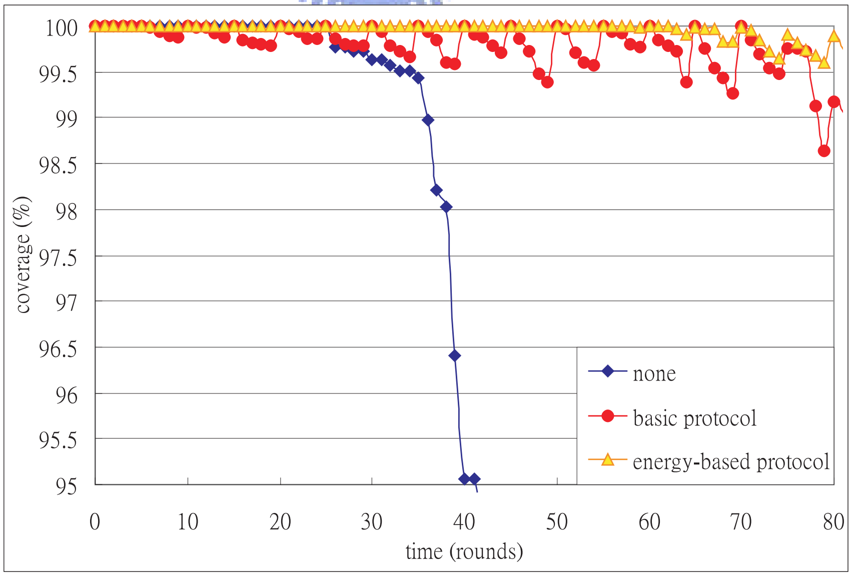


Figure 6.5: The ratio of covered area.

curves “grid_1”, “ grid_5”, and “grid_10” indicate the performances of the scheme in [40] with different grid sizes, 1×1, 5×5, and 10×10, respectively. The curve “none” is the result of that all sensors keep awake all the time and our basic scheme presented in Section 6.2.1 is labeled by “basic protocol”. As aforementioned, grid approximation may cause inaccuracy. So, as can be seen in the figure, only relatively smaller grid size, 1×1, can achieve almost 100% coverage. By contrast, our basic scheme can easily achieve the similar coverage. Note that the vibrations in the figure are resulted from some sensors may run out its energy during a cycle.

Next, we compare our basic scheme with the energy-based scheduling. There are two kinds of performances evaluated, the ratio of alive nodes and the ratio of covered area. The results are shown in Fig. 6.4 and Fig. 6.5. As can be seen in Fig. 6.4, adopting our basic scheme can keep all sensors alive much longer than turning each sensor always on. In addition, our energy-based scheme further outperform the basic scheme. Similar results can also be seen on the ratio of covered area, as shown in Fig. 6.5. The energy-based scheduling can keep almost 100% coverage until about 60 rounds while the curve of the basic scheme vibrates from about 10 rounds.

Chapter 7

Conclusions and Future Works

In this dissertation, we have defined and proposed solutions to the coverage problems both in 2D and 3D spaces for wireless sensor networks. We model the coverage problem as a decision problem, whose goal is to determine whether each location of the target sensing area is sufficiently covered or not. Furthermore, we have studied the relationship between coverage and connectivity, and proposed more general solutions. In addition, we have presented decentralized coverage-preserving node-scheduling protocols to prolong network time, which can significantly reduce the computational complexity incurred on each sensor.

For the two-dimensional coverage problem, rather than determining the level of coverage of each location, our solutions are based on checking the perimeter of each sensor's sensing range. Although the problem seems to be very difficult at the first glance, our scheme can give an exact answer in $O(nd \log d)$ time. With the proposed techniques, we also discuss several applications (such as discovering insufficiently covered regions and saving energies) and extensions (such as scenarios with hot spots and irregular sensing ranges) of our results. A software tool that implements the proposed algorithms is available on the web (http://hscc.csie.nctu.edu.tw/download/coverage.zip) for free download.

We have also proposed a solution to the three-dimensional coverage problem for wireless sensor networks. We have shown that tackling this problem in a 3D space can be done at polynomial time. Our solution reduces the geometric problem from a 3D space to a 2D space, and further to a 1D space, thus leading to a very efficient solution.

Next, the relationship between sensing coverage and communication connectivity of a sensor network is studied. Solutions are proposed to determine whether a network is sufficiently covered and connected. Contrary to existing works, we make no assumption on the relationship between communication distances and sensing distances. Our approach looks at how the perimeter of each sensor's sensing range is covered by its neighbors, and whether there exists a link/path to each of its neighbors, thus leading to very efficient solutions. We show conditions for a sensor network to be k-covered and k-connected, and to be k-covered and 1-connected. We also show under what conditions a sensor network may be decomposed into multiple connected components each of which provides sufficient coverage. Distributed protocols to determine coverage and connectivity of a network is then discussed.

Finally, decentralized energy-conserving and coverage-preserving protocols targeted at extending the network lifetime are presented. The proposed protocol is similar to the model in [40]. However, intersection points between sensors' sensing ranges are used to ensure the network coverage, which can significantly reduce the computational complexity incurred on each sensor and completely eliminate the inaccuracy problem caused by gird approximation. Besides, we further discuss how to utilize sensors' remaining energy to adjust parameters in our protocols to balance the energy consumption among sensors. Through simulation studies, the energy-based parameter settings are shown to outperform the basic scheme.

For the future, we try to utilize the result of the 3D coverage problem in deploying sensors in 3D space and in reducing on-duty time of wireless sensors. Besides, we are currently investigating the possibility of applying our result to control the level of coverage and connectivity of a network. Since sensors may be deployed in an arbitrary manner, redundant nodes may exist. If the level of coverage is more than needed, we can properly schedule nodes' on-duty time to reduce the level of coverage, and thus prolong the network lifetime. This is the same for network connectivity. Further, transmission power control can be integrated into the mechanism. Next, we intend to extend the energy-conserving and coverage-preserving protocols to ensure the network to be k-covered. We will report the related results in our future papers.

Bibliography

[1] S. Adlakha and M. Srivastava. Critical density thresholds for coverage in wireless sensor networks. In *IEEE Wireless Communications and Networking Conf. (WCNC)*, pages 1615 – 1620, 2003.

[2] P. K. Agarwal and M. Sharir. Arrangements and their applications. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 49–119. Elsevier, North-Holland, New York, 2000.

[3] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM*, pages 775–784, 2000.

[4] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, pages 22–31, 2002.

[5] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Commun.*, 7(5):28–34, Oct. 2000.

[6] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does topology control reduce interference? In *ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 9 – 19, 2004.

[7] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *ACM/Kluwer Wireless Networks*, 8(5):481–494, Sep. 2002.

[8] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly resilient, energy efficient multipath routing in wireless sensor networks. *ACM Mobile Comput. and Commun. Review*, 5(4):11–25, Oct. 2001.

[9] H. Gupta, S. R. Das, and Q. Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 189–200, 2003.

[10] P. Hall. *Introduction to the Theory of Coverage Processes*. Wiley, New York, 1988.

[11] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 21, pages 389–412. CRC Press LLC, Boca Raton, FL, 1997.

[12] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Hawaii Int'l Conf. on Systems Science (HICSS)*, pages 3005–3014, 2000.

[13] A. Heppes and J. B. M. Melissen. Covering a rectangle with equal circles. *Period. Math. Hung.*, 34:65–81, 1996.

[14] Q. Huang. Solving an open sensor exposure problem using variational calculus. Technical Report WUCS-03-1, Washington University, Department of Computer Science and Engineering, St. Louis, Missouri, 2003.

[15] N. Li and J. C. Hou. FLSS: A fault-tolerant topology control algorithm for wireless networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, 2004.

[16] X.-Y. Li, P.-J. Wan, and O. Frieder. Coverage in wireless ad hoc sensor networks. *IEEE Trans. Comput.*, 52(6):753–763, June 2003.

[17] J. Lu and T. Suda. Coverage-aware self-scheduling in sensor networks. In *IEEE Computer Communications Workshop (CCW)*, pages 117 – 123, 2003.

[18] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. B. Srivastava. Coverage problems in wireless ad-hoc sensor networks. In *IEEE INFOCOM*, pages 1380–1387, 2001.

[19] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure in wireless ad-hoc sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 139–150, 2001.

[20] S. Meguerdichian, S. Slijepcevic, V. Karayan, and M. Potkonjak. Localized algorithms in wireless ad-hoc networks: location discovery and sensor exposure. In *ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 106–116, 2001.

[21] J. B. M. Melissen and P. C. Schuur. Improved coverings of a square with six and eight equal circles. *Electronic Journal of Combinatorics*, 3(1), 1996.

[22] D. Nicules and B. Nath. Ad-hoc positioning system (APS) using AoA. In *IEEE INFOCOM*, pages 1734–1743, 2003.

[23] K. J. Nurmela and P. R. J. Östergård. Covering a square with up to 30 equal circles. Research Report A62, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland, June 2000.

[24] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Aug. 1987.

[25] J. O'Rourke. Computational geometry column 15. *Int'l Journal of Computational Geometry and Applications*, 2(2):215–217, 1992.

[26] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, May 2000.

[27] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 166–179, 2001.

[28] S. Shakkottai, R. Srikant, and N. Shroff. Unreliable sensor grids: coverage, connectivity and diameter. In *IEEE INFOCOM*, pages 1073 – 1083, 2003.

[29] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 272–287, 2001.

[30] S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE Int'l Conf. on Communications (ICC)*, pages 472–476, 2001.

[31] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Commun.*, 7(5):16–27, Oct. 2000.

[32] D. Tian and N. D. Georganas. A node scheduling scheme for energy conservation in large wireless sensor networks. *Wireless Commun. and Mobile Comput. (WCMC)*, 3:271–290, 2003.

[33] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Int'l Workshop on Information Processing in Sensor Networks (IPSN)*, 2003.

[34] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. Minimal and maximal exposure path algorithms for wireless embedded sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 40–50, 2003.

[35] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless, and C. Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 28–39, 2003.

[36] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *IEEE INFOCOM*, pages 1388 – 1397, 2001.

[37] R. Williams. *The Geometrical Foundation of Natural Structure: A Source Book of Design*, pages 51–52. Dover, New York, 1979.

[38] A. Woo and D. E. Culler. A transmission control scheme for media access in sensor networks. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 221–235, 2001.

[39] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *ACM Int'l Conf. on Mobile Computing and Networking (MobiCom)*, pages 70–84, 2001.

[40] T. Yan, T. He, and J. A. Stankovic. Differentiated surveillance for sensor networks. In *ACM Int'l Conf. on Embedded Networked Sensor Systems (SenSys)*, pages 51–62, 2003.

[41] F. Ye, G. Zhong, S. Lu, and L. Zhang. PEAS: A robust energy conserving protocol for long-lived sensor networks. In *Int'l Conf. on Distributed Computing Systems (ICDCS)*, pages 28 – 37, 2003.

[42] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *IEEE INFOCOM*, pages 1567–1576, 2002.

[43] H. Zhang and J. C. Hou. Maintaining sensing coverage and connectivity in large sensor networks. In *NSF International Workshop on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, 2004.

Publication List

Journal Paper

1. C.-F. Huang and Y.-C. Tseng, "On Topology Improvement of a Packet Radio Network by Power Control", ***IEEE Transactions on Vehicular Technology***, Vol. 52, No. 4, July 2003, pp. 985-998.
2. C.-F. Huang, H.-W. Lee, and Y.-C. Tseng, "A Two-Tier Heterogeneous Mobile Ad Hoc Network Architecture and Its Load-Balance Routing Problem", ***ACM/Kluwer Mobile Networking and Applications (MONET)***, Vol. 9, No. 4, Aug. 2004, pp. 379-391, Special Issue on Integration of Heterogeneous Wireless Technologies.
3. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, "Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies", ***The Computer Journal*** (to appear).
4. C.-F. Huang and Y.-C. Tseng, "The Coverage Problem in a Wireless Sensor Network", ***ACM/Kluwer Mobile Networking and Applications (MONET)***, Special Issue on Wireless Sensor Networks (accepted).
5. C.-F. Huang and Y.-C. Tseng, "A Survey of Solutions to the Coverage Problems in Wireless Sensor Networks", ***Journal of Internet Technology***, Special Issue on Wireless Ad Hoc and Sensor Networks, Mar. 2004 (accepted).

Conference Paper

1. C.-F. Huang, Y.-C. Tseng, S.-L. Wu, and J.-P. Sheu, "Increasing the Throughput of Multihop Packet Radio Networks with Power Adjustment", ***Int'l Conf. on Computer Communication and Networks (ICCCN)***, 2001.
2. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, "A Mobile-Agent Approach for Location Tracking in a Wireless Sensor Network", ***Int'l Computer Symp. (ICS)***, 2002.
3. Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang, "Location Tracking in a Wireless Sensor Network by Mobile Agents and Its Data Fusion Strategies", ***Int'l Workshop on Information Processing in Sensor Networks (IPSN)***, 2003.
4. C.-F. Huang, H.-W. Lee, and Y.-C. Tseng, "A Two-Tier Heterogeneous Mobile Ad Hoc Network Architecture and Its Load-Balance Routing Problem", ***IEEE Semiannual Vehicular Technology Conference (VTC) Fall***, 2003.

5. C.-F. Huang and Y.-C. Tseng, “The Coverage Problem in a Wireless Sensor Network”, ***ACM Int’l Workshop on Wireless Sensor Networks and Applications (WSNA)*** (in conjunction with ACM MobiCom), 2003.
6. C.-F. Huang, Y.-C. Tseng, and Li-Chu Lo, “The Coverage Problem in Three-Dimensional Wireless Sensor Networks”, ***IEEE GLOBECOM***, 2004.
7. Y.-Y. Hsu, Y.-C. Tseng, C.-C. Tseng, C.-F. Huang, J.-H. Fan, and H.-L. Wu, “Design and Implementation of Two-tier Mobile Ad Hoc Networks with Seamless Roaming and Load-balancing Routing Capacity”, ***Int’l Conf. on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)***, 2004

Book Chapters

1. Y.-C. Tseng, C.-F. Huang and, S.-P. Kuo, “Positioning and Tracking in Wireless Sensor Networks” (a book chapter in ***Handbook of Sensor Networks***, CRC Press)
2. C.-F. Huang, P.-Y. Chen, Y.-C. Tseng, and W.-T. Chen, "Models and Algorithms for Coverage Problems in Wireless Sensor Networks" (a book chapter in ***Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Networks***, CRC Press, edited by J. Wu, expected 2004).

Submitted Paper

1. Chi-Fu Huang, Yu-Chee Tseng, and Hsiao-Lu Wu, “Ensuring Both Coverage and Connectivity of a Wireless Sensor Network”, ***IEEE Int’l Conf. on Pervasive Computing and Communications (PerCom’05)***, submitted, Aug., 2004.