2

Sensor Node Deployment

An important objective of sensor networks is to effectively monitor the environment, detect, localize, and classify targets of interest. The effectiveness of these networks is determined to a large extent by the coverage provided by the sensor deployment. The positioning of sensors affects coverage, communication cost, and resource management. In this chapter[1], we investigate two important aspects of sensor node deployment in wireless sensor networks [134, 135]. We introduce the virtual force algorithm (VFA) for sensor deployment in Section 2.2, which is a fast algorithm with low computation overhead that improves the coverage for a given number of sensors within a cluster in cluster-based sensor networks. In Section 2.4, we study the problem of uncertainty modeling for sensor node deployment in wireless sensor networks.

2.1 Sensor Node Detection Models

The sensor field is represented by a two-dimensional grid. The dimensions of the grid provide a measure of the sensor field. The granularity of the grid, i.e. distance between grid points can be adjusted to trade off computation time of the VFA algorithm with the effectiveness of the coverage measure. The detection by each sensor is modeled as a circle on the two-dimensional grid. The center of the circle denotes the sensor while the radius denotes the detection range of the sensor. We first consider a binary detection model in which a target is detected (not detected) with complete certainty by the sensor if a target is inside (outside) its circle. The binary model facilitates the understanding of the VFA model. We then investigate two types of realistic probabilistic models in which the probability that the sensor detects a target depends on the relative position of the target.

[1] This chapter is based on Y. Zou and K. Chakrabarty, "Sensor deployment and target localization in distributed sensor networks", *ACM Transactions on Embedded Computing Systems*, vol. 3, pp. 61-91, February 2004.

Let us consider a sensor field represented by a $m \times n$ grid. Let s be an individual sensor node on the sensor field located at grid point (x, y). Each sensor node has a detection range of r. For any grid point P at (i, j), we denote the Euclidean distance between s at (x, y) and P at (i, j) as $d_{ij}(x, y)$, i.e. $d_{ij}(x, y) = \sqrt{(x-i)^2 + (y-j)^2}$. Equation (2.1) shows the binary sensor model [23] that expresses the coverage $c_i j(x, y)$ of a grid point at (i, j) by sensor s at (x, y).

$$c_{ij}(x, y) = \begin{cases} 1, & \text{if } d_{ij}(x, y) < r \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The binary sensor model assumes that sensor readings have no associated uncertainty. In reality, sensor detections are imprecise, hence the coverage $c_{ij}(x, y)$ needs to be expressed in probabilistic terms. A possible way of expressing this uncertainty is to assume the detection probability on a target by a sensor varies exponentially with the distance between the target and the sensor [33, 34]. This probabilistic sensor detection model given in Equation (2.2).

$$c_{ij}(x, y) = e^{-\alpha d_{ij}(x,y)} \tag{2.2}$$

This is also the coverage confidence level of this point from sensor s. The parameter α can be used to model the quality of the sensor and the rate at which its detection probability diminishes with distance. Clearly, the detection probability is 1 if the target location and the sensor location coincide. Alternatively, we can also use another probabilistic sensor detection model given in Equation (2.3), which is motivated in part by [36].

$$c_{ij}(x, y) = \begin{cases} 0, & \text{if } r + r_e \leq d_{ij}(x, y) \\ \mathrm{e}^{-\lambda a^{\beta}}, & \text{if } r - r_e < d_{ij}(x, y) < r + r_e \\ 1, & \text{if } r - r_e \geq d_{ij}(x, y) \end{cases} \tag{2.3}$$

Note that $r_e (r_e < r)$ is a measure of the uncertainty in sensor detection, $a = d_{ij}(x, y) - (r - r_e)$, and λ and β are parameters that measure detection probability when a target is at distance greater than r_e but within a distance from the sensor. This model reflects the behavior of range sensing devices such as infrared and ultrasound sensors. The probabilistic sensor detection model is shown in Figure 2.1. Note that distances are measured in units of grid points. Figure 2.1 also illustrates the translation of a distance response from a sensor to the confidence level as a probability value about this sensor response. Different values of the parameters α and β yield different translations reflected by different detection probabilities, which can be viewed as the characteristics of various types of physical sensors.

It is often the case that there are obstacles in the sensor field terrain. If we are provided with such *a priori* knowledge about where obstacles in the sensor field, we can also build the terrain information into our models based on the principle of line of sight. An example is given in Figure 2.2. Some types of sensors are not able to see through any obstacles located in

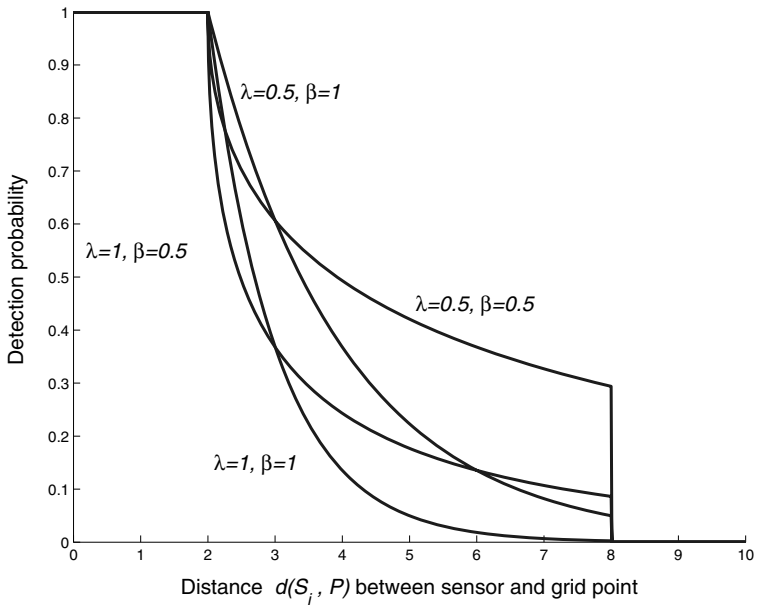

Fig. 2.1. Probabilistic sensor detection model.

the sensor field; hence models and algorithms must consider the problem of achieving an adequate sensor field coverage in presence of obstacles. Suppose C_{xy} is a $m \times n$ matrix that corresponds to the detection probabilities of each grid point in the sensor field when a sensor node is located at grid point (x, y), i.e., $C_{xy} = [c_{ij}(x, y)]_{m \times n}$. To achieve the coverage in presence of obstacles, we need to generate a mask matrix for the corresponding coverage probability matrix C_{xy} to mask out those grid points as the "blocked area", as shown in Figure 2.2. In this way, the sensor node placed at the location (x, y) will not see any grid points beyond the obstacles. We also assume that sensor nodes are not placed on any grid points with obstacles. Figure 2.3 is an example of the mask matrix for a sensor node at $(1, 1)$ in a 10 by 10 sensor field grid with obstacles located at $(7, 3), (7, 4), (3, 5), (4, 5), (5, 5)$.

2.2 Virtual Force Algorithm

As an initial sensor node deployment step, a random placement of sensors in the target area (sensor field) is often desirable, especially if no *a priori* knowledge of the terrain is available. Random deployment is also practical in military applications, where wireless sensor networks are initially established by dropping or throwing sensors into the sensor field. However, random deployment does not always lead to effective coverage, especially if the sensors are overly clustered and there is a small concentration of sensors in certain

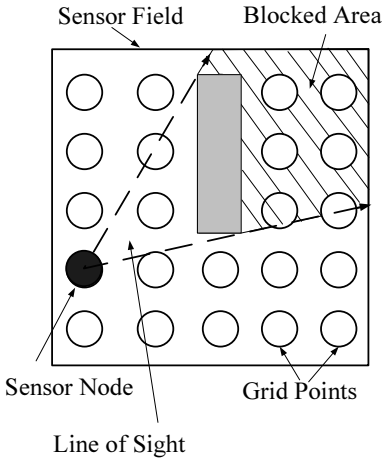

Fig. 2.2. Example to illustrate the line of sight principle.

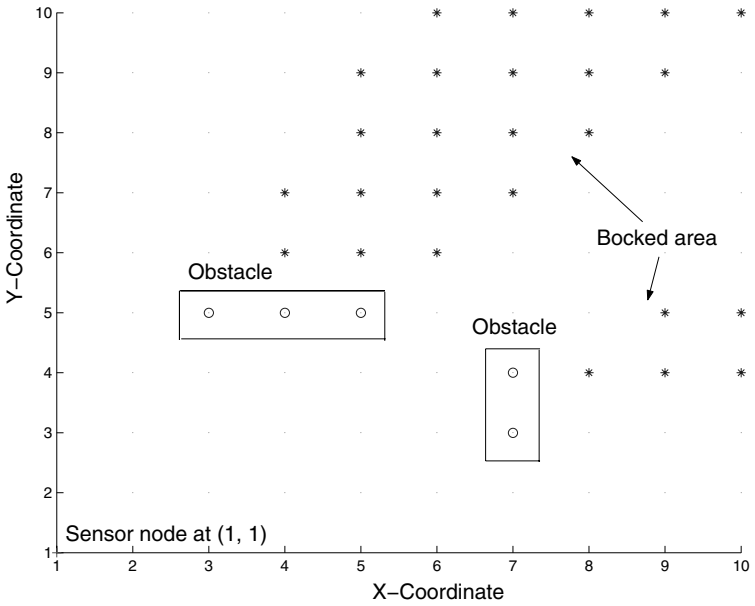

Fig. 2.3. Obstacle mask matrix example.

parts of the sensor field. However the coverage provided by a random deployment can be improved using a force-directed algorithm. We present the virtual force algorithm (VFA) as a sensor deployment strategy to enhance the coverage after an initial random placement of sensors. The VFA algorithm combines the ideas of potential field [28, 57, 79] and disk packing [78]. For a given number of sensors, VFA attempts to maximize the sensor field coverage

using a combination of attractive and repulsive forces. During the execution of the force-directed VFA algorithm, sensors do not physically move but a sequence of virtual motion paths is determined for the randomly-placed sensors. Once the effective sensor positions are identified, a one-time movement is carried out to redeploy the sensors at these positions. Energy constraints are also included in the sensor repositioning algorithm. In the sensor field, each sensor behaves as a "source of force" for all other sensors. This force can be either positive (attractive) or negative (repulsive). If two sensors

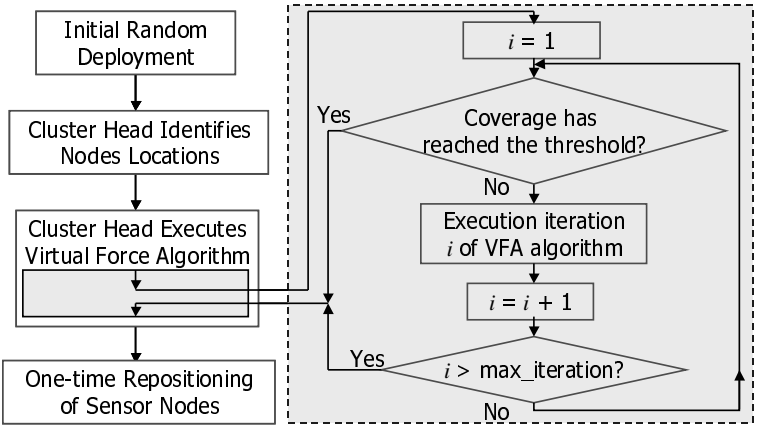

Fig. 2.4. Sensor deployment with VFA algorithm.

are placed too close to each other, the "closeness" being measured by a pre-determined threshold, they exert negative forces on each other. This ensures that the sensors are not overly clustered, leading to poor coverage in other parts of the sensor field. On the other hand, if a pair of sensors is too far apart from each (once again a pre-determined threshold is used here), they exert positive forces on each other. This ensures that a globally uniform sensor placement is achieved. Figure 2.4 illustrates how VFA algorithm is used for sensor deployment.

2.2.1 Virtual Forces

We now describe the virtual forces and virtual force calculation in the VFA algorithm. In the following discussion, we use the notation introduced in the previous subsection. Let S denote the set of deployed sensors node, i.e., $S = \{s_1, \ldots, s_k\}$ and $|S| = k$. Let the total virtual force action on a sensor node $s_p (p = 1, \ldots, k)$ be denoted by $\mathbf{F}_p$. Note that $\mathbf{F}_p$ is a vector whose orientation is determined by the vector sum of all the forces acting on s_p. Let the force exerted on s_p by another sensor $s_q (q = 1, \ldots, k, q \neq p)$ be denoted by $\mathbf{F}_{pq}$. In addition to the positive and negative forces due to other sensors, a sensor s_p is

also subjected to forces exerted by obstacles and areas of preferential coverage in the grid. This provides us with a convenient method to model obstacles and the need for preferential coverage. Sensor deployment must take into account the nature of the terrain, e.g., obstacles such as building and trees in the line of sight for infrared sensors, uneven surface and elevations for hilly terrain, etc. In addition, based on relative measures of security needs and tactical importance, certain areas of the grid need to be covered with greater certainty.

The knowledge of obstacles and preferential areas implies a certain degree of *a priori* knowledge of the terrain. In practice, the knowledge of obstacles and preferential areas can be used to direct the initial random deployment of sensors, which in turn can potentially increase the efficiency of the VFA algorithm. In our virtual force model, we assume that obstacles exert repulsive (negative) forces on a sensor. Likewise, areas of preferential coverage exert attractive (positive) forces on a sensor. If more detailed information about the obstacles and preferential coverage areas is available, the parameters governing the magnitude and direction (i.e., attractive or repulsive) of these forces can be chosen appropriately. In this work, we let $\mathbf{F}_{pA}$ be the total attractive force on s_p due to preferential coverage areas, and let $\mathbf{F}_{pR}$ be the total repulsive force on s_p due to obstacles. The total force $\mathbf{F}_p$ on s_p can now be expressed as,

$$\mathbf{F}_p = \sum_{q=1,\ q\neq p}^{k} \mathbf{F}_{pq} + \mathbf{F}_{pR} + \mathbf{F}_{pA} \tag{2.4}$$

We next express the force $\mathbf{F}_{pq}$ between s_p and s_q in polar coordinate notation. Note that $\mathbf{f} = (r, \theta)$ implies a magnitude of r and orientation θ for vector $\mathbf{f}$.

$$\mathbf{F}_{pq} = \begin{cases} (w_A(d_{pq} - d_{th}), \theta_{pq}) & \text{if } d_{pq} > d_{th} \\ 0, & \text{if } d_{pq} = d_{th} \\ (w_R \frac{1}{d_{pq}}, \theta_{pq} + \pi), & \text{if otherwise} \end{cases} \tag{2.5}$$

where $d_{pq} = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$ is the Euclidean distance between sensor s_p and s_q, d_{th} is the threshold on the distance between s_p and s_q, θ_{pq} is the orientation (angle) of a line segment from s_p to s_q, and $w_A(w_R)$ is a measure of the attractive (repulsive) force. The threshold distance d_{th} controls how close sensors get to each other. As an example, consider the four sensors s_1, s_2, s_3 and s_4 in Figure 2.5. The force $\mathbf{F}_1$ on s_1 is given by $\mathbf{F}_1 = \mathbf{F}_{12} + \mathbf{F}_{13} + \mathbf{F}_{14}$. If we assume that $d_{12} > d_{th}$, $d_{13} < d_{th}$, and $d_{14} = d_{th}$, s_2 exerts an attractive force on s_1, s_3 exerts a repulsive force on s_1 and s_4 exerts no force on s_1. This is shown in Figure 2.5. Note that d_{th} is a pre-determined parameter that is supplied by the user, who can choose an appropriate value of d_{th} to achieve a desired coverage level over the sensor field.

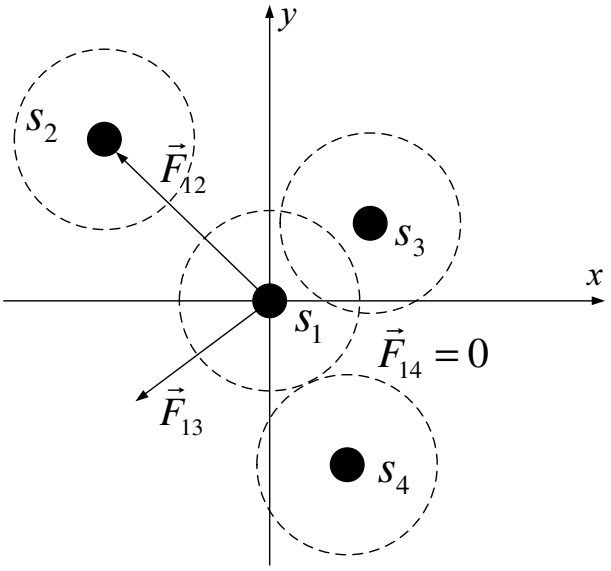

Fig. 2.5. An example of virtual forces with four sensors.

2.2.2 Overlapped Sensor Detection Areas

If $r_e \approx 0$ and we use the binary sensor detection model given by Equation (2.1), we attempt to make d_{pq} as close to $2r$ as possible. This ensures that the detection regions of two sensors do not overlap, thereby minimizing "wasted overlap" and allowing us to cover a large grid with a small number of sensors. This is illustrated in Figure 2.6(a). An obvious drawback here is that a few grid points are not covered by any sensor. Note that an alternative strategy is to allow overlap, as shown in Figure 2.6(b). While this approach ensures that all grid points are covered, it needs more sensors for grid coverage. Therefore, we adopt the first strategy. Note that in both cases, the coverage is effective only if the total area $k\pi r^2$ that can be covered with the k sensors exceeds the area of the grid.

If $r_e > 0$, r_e is not negligible and the probabilistic sensor model given by Equation (2.2) or Equation (2.3) is used. Note that due to the uncertainty in sensor detection responses, grid points are not uniformly covered with the same probability. Some grid points will have low coverage if they are covered only by only one sensor and they are far from the sensor. In this case, it is necessary to overlap sensor detection areas in order to compensate for the low detection probability of grid points that are far from a sensor. Consider a grid point with coordinate (i, j) lying in the overlap region of sensors s_p and s_q located at (x_p, y_p) and (x_q, y_q) respectively. Let $c_{ij}(s_p, s_q)$ be the probability that a target at this grid point is reported as being detected by observing the outputs of these two sensors. We assume that sensors within a cluster operate

independently in their sensing activities. Thus

$$c_{ij}(s_p, s_q) = 1 - (1 - c_{ij}(s_p))(1 - c_{ij}(s_q)) \tag{2.6}$$

where $c_{ij}(s_p) = c_{ij}(x_p, y_p)$ and $c_{ij}(s_q) = c_{ij}(x_q, y_q)$ are coverage probabilities from the probabilistic sensor detection models as we defined in Section 2.1. Since the term $1 - (1 - c_{ij}(s_p))(1 - c_{ij}(s_q))$ expresses the probability that neither s_p nor s_q covers grid point at (i, j), the probability that the grid point (i, j) is covered is given by Equation (2.6). Let c_{th} be the desired coverage threshold for all grid points. This implies that

$$\min_{i,j}\{c_{ij}(s_p, s_q)\} \geq c_{th} \tag{2.7}$$

Note that Equation (2.6) can also be extended to a region which is overlapped by a set of k_{ov} sensors, denoted as S_{ov}, $k_{ov} = |S_{ov}|$, $S_{ov} \subseteq \{s_1, s_2, \ldots, s_k\}$. The coverage of the grid point at (i, j) due to a set of sensor nodes S_{ov} in this case is given by:

$$c_{ij}(S_{ov}) = 1 - \prod_{s_p \in S_{ov}} (1 - c_{ij}(s_p)) \tag{2.8}$$

As shown in Equation (2.5), the threshold distance d_{th} is used to control how close sensors get to each other. When sensor detection areas overlap, the closer the sensors are to each other, the higher is the coverage probability for grid points in the overlapped areas. Note however that there is no increase in the point coverage once one of the sensors gets close enough to provide detection with a probability of one. Therefore, we need to determine d_{th} that maximizes the number of grid points in the overlapped area that satisfies $c_{ij}(s_p) > c_{th}$. Let us consider the three sensors s_1, s_2, and s_3 in Figure 2.6(a), where no overlap exists. Assume the three sensors are on a 31 by 31 grid, $r = 5$ and $r_e = 3$ in units of grid points. Figures 2.7–2.9 show how the coverage is affected by d_{th} and c_{th} when the threshold distance d_{th} is changed from $r + r_e$ to $r - r_e$. The coverage for the entire grid is calculated as the fraction of grid points that exceeds the threshold c_{th}. We can use these graphs to appropriately choose d_{th} according to the required c_{th}.

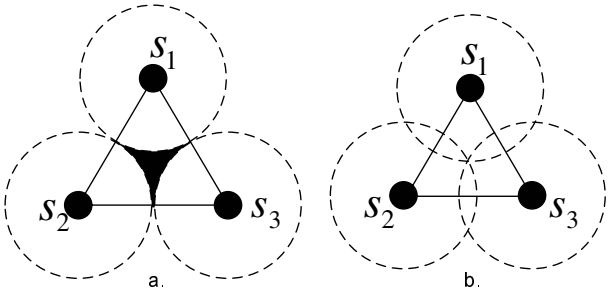

Fig. 2.6. Non-overlapped and overlapped sensor coverage areas.

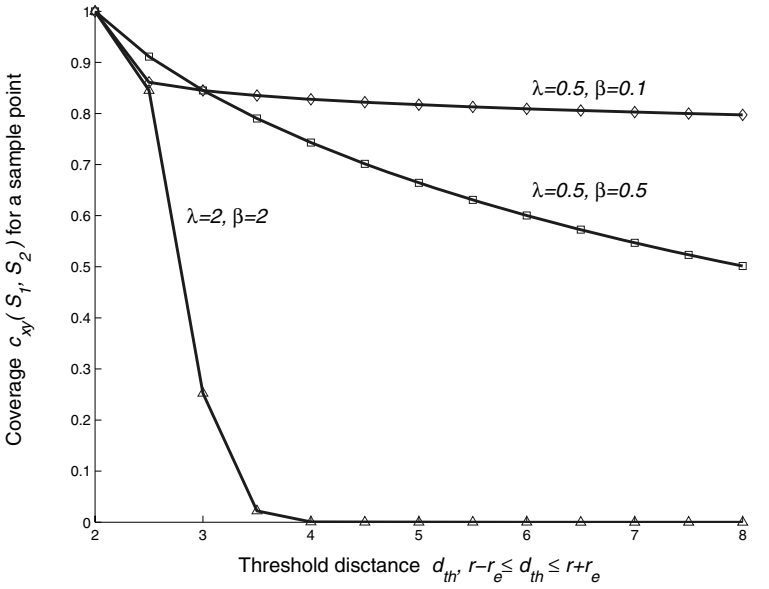

Fig. 2.7. Coverage vs. d_{th} of a sample point inside the overlapped area of s_1 and s_2.

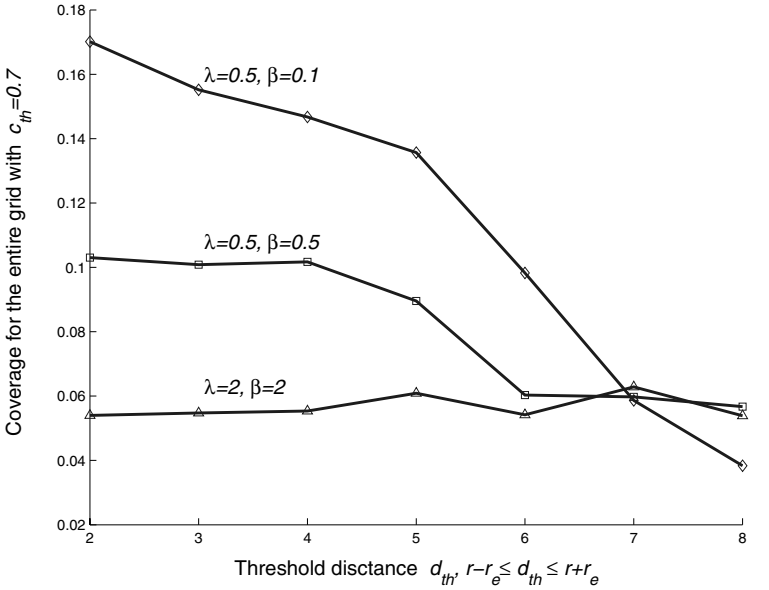

Fig. 2.8. Coverage vs. d_{th} with $c_{th} = 0.7$ and different λ and β.

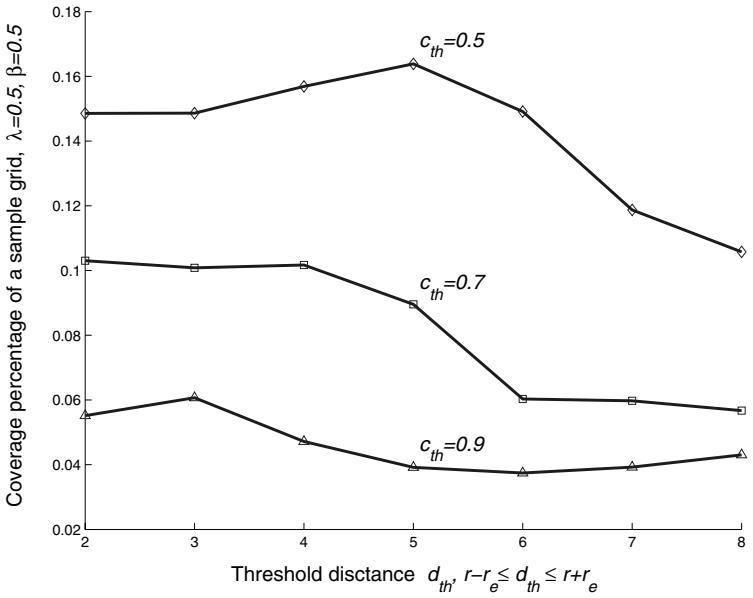

Fig. 2.9. Coverage vs. d_{th} with $\lambda = 0.5$ and $\beta = 0.5$ and $c_{th} = 0.5$, 0.7 and 0.9.

2.2.3 Energy Constraint on the VFA Algorithm

In order to prolong the battery life, the distances between the initial and final position of the sensors are limited in the repositioning phase to conserve energy. We use $d_{max}(s_p)$ to denote the maximum distance that sensor s_p can move in the repositioning phase. To simplify the discussion without loss of generality, we assume $d_{max}(s_p) = d_{max}(s_q) = d_{max}$, for $p, q = 1, 2, \ldots, k$. During the execution of the VFA algorithm, for each sensor node, whenever the distance from the current virtual position to the initial position reaches the distance limit d_{max}, any virtual forces on this sensor are disabled. For sensor s_p, let $(x_p, y_p)_{random}$ be the initial location obtained from the random deployment, and $(x_p, y_p)_{virtual}$ be the location generated by the VFA algorithm. The energy constraint can be described as:

$$\mathbf{F}_p = \begin{cases} 0, & \text{if } d((x_p, y_p)_{rand}, (x_p, y_p)_{virtual}) \geq d_{max} \\ \mathbf{F}_p, & \text{otherwise (i.e., the force is unchanged)} \end{cases} \tag{2.9}$$

Therefore the virtual force $\mathbf{F}_p$ given by Equation (2.4) on sensor s_p is ignored whenever the move violates the energy constraint expressed by d_{max}. Note that due to the energy constraint on the one-time repositioning given by Equation (2.9), it might be necessary to trade off the coverage with the energy consumed in repositioning if d_{max} is not large enough.

Note that the VFA algorithm is designed to be executed on the cluster head, which is expected to have more computational capabilities than sensor

nodes. The cluster head uses the VFA algorithm to find appropriate sensor node locations based on the coverage requirements. The new locations are then sent to the sensor nodes, which perform a one-time movement to the designated positions. No movements are performed during the execution of the VFA algorithm.

2.2.4 Procedural Description

We next describe the VFA algorithm in pseudo-code. Figure 2.10 shows the data structure of the VFA algorithm and Figure 2.11 shows the implementation details in pseudo code form. For a n by m grid with a total of k sensors deployed, the computational complexity of the VFA algorithm is $O(nmk)$. Due to the granularity of the grid and the fact that the actual coverage is evaluated by the number of grid points that have been adequately covered, the convergence of the VFA algorithm is controlled by a threshold value, denoted by Δc. Let us use $c(loops)$ to denote the current grid coverage of the number *loops* iteration in the VFA algorithm. For the binary sensor detection

VFA Data Structures: $\texttt{Grid}, \{\texttt{s}_1, \texttt{s}_2, \ldots, \texttt{s}_\texttt{k}\}$

/* n_P is the number of preferential area blocks (attractive forces) and n_O is the number of obstacle blocks (repulsive forces). S_{ij}, k_{ij} and p_table_{ij} are used for energy-aware target localization described in Chapter 3.
$(x, y)_{VFA}$ is the final position found by the VFA algorithm. d_{max} is the energy constraint on the sensor re-positioning phase in the VFA algorithm. */
1 *Grid* structure:
2 Properties: *width*, *height*, k, c_{th}, d_{th}, $c(loops)$, $\bar{c}$, Δc;
3 Preferential areas: $PA_i(x, y, wx, wy)$, $i = 1, 2, \ldots, n_P$;
4 Obstacles areas: $OA_i(x, y, wx, wy)$, $i = 1, 2, \ldots, n_O$;
5 Grid points, P_{ij}: $c_{ij}(\{s_1, s_2, \ldots, s_k\})$, $S_{ij}, k_{ij}, p\_table_{ij}$;
6 Sensor s_p structure:
$(x_p, y_p)_{random}, (x_p, y_p)_{virtual}, (x, y)_{VFA}, p, r, r_e, \alpha, \beta, d_{max}$;

Fig. 2.10. Data structures used in the VFA algorithm.

model without the energy constraint, the upper bound value denoted as $\bar{c}$ is $k\pi r^2$; for the probabilistic sensor detection model or binary sensor detection model with the energy constraint, $c(loops)$ is checked for saturation by defining $\bar{c}$ as the average of the coverage ratios of the near 5 (or 10) iterations. Therefore, the VFA algorithm continues to iterate until $|c(loops) - \bar{c}| \leq \Delta c$. In our experiments, Δc is set to 0.001.

Note that there exists the possibility of certain pathological scenarios in which the VFA algorithm is rendered ineffective, e.g., if the sensors are initially placed along the circumference of a circle such that all virtual forces are

Procedure *Virtual_Force_Algorithm* ($\mathtt{Grid}, \{\mathtt{s}_1, \mathtt{s}_2, \ldots, \mathtt{s}_k\}$)

1 Set $loops = 0$;
2 Set $MaxLoops$ =**MAX_LOOPS**;
3 **While** ($loops < MaxLoops$)
4 /* coverage evaluation */
5 **For** grid point P at (i, j) in Grid, $i \in [1, width], j \in [1, height]$
6 **For** $s_p \in \{s_1, s_2, \ldots, s_k\}$
7 Calculate $c_{ij}(x_p, y_p)$ from the sensor model using $(d_{ij}(x_p, y_p), c_{th}, d_{th}, \alpha, \beta)$;
8 **End**
9 **End**
10 **If** coverage requirements are met: $|c(loops) - \bar{c}| \leq \Delta c$
11 **Break** from **While** loop;
12 **End**
13 /* virtual forces among sensors */
14 **For** $s_p \in \{s_1, s_2, \ldots, s_k\}$
15 Calculate $\mathbf{F}_{pq}$ using $d(s_p, s_q)$, d_{th}, w_A, w_R;
16 Calculate $\mathbf{F}_{pA}$ using $d(s_p, PA_1, \ldots, PA_{n_P})$, d_{th};
17 Calculate $\mathbf{F}_{pR}$ using $d(s_p, OA_1, \ldots, OA_{n_O})$, d_{th};
18 $\mathbf{F}_p = \sum \mathbf{F}_{pq} + \mathbf{F}_{pR} + \mathbf{F}_{pA}, q = 1, \ldots, k, q \neq p$;
19 **End**
20 /*move sensors virtually */
21 **For** $s_p \in \{s_1, s_2, \ldots, s_k\}$
22 /* energy constraint on the sensor movement */
23 **If** $d((x_p, y_p)_{random}, (x, y)_{virtual}) \geq d_{max}$
24 Set $\mathbf{F}_p = 0$;
25 **End**
26 $\mathbf{F}_p$ virtually moves s_p to its next position;
27 **End**
28 /* continue to next iteration */
29 Set $loops = loops + 1$;
30 **End**

Fig. 2.11. Pseudocode of the VFA algorithm.

balanced. The efficiency of the VFA algorithm depends on the values of the force parameters w_A and w_R. We found that the algorithm converged more rapidly for our case studies if $w_R \gg w_A$. This need not always be true, so we are examining ways to choose appropriate values for w_R and w_A based on the initial configuration.

2.3 VFA Simulation Results

In this section, we present simulation results obtained using the VFA algorithm. The deployment requirements include the maximum improvement of coverage over random deployment, the coverage for preferential areas and the avoidance of obstacles. For all simulation results presented in this section, distances are measured in units of grid points. A total of 20 sensors are placed in the sensor field in the random placement stage. Each sensor has a detection radius of 5 units ($r = 5$), and range detection error of 3 units ($r_e = 3$) for the probabilistic detection model. The sensor field is 50 by 50 in dimension. The simulation is done on a Pentium III 1.0GHz PC using Matlab.

2.3.1 Case Study 1

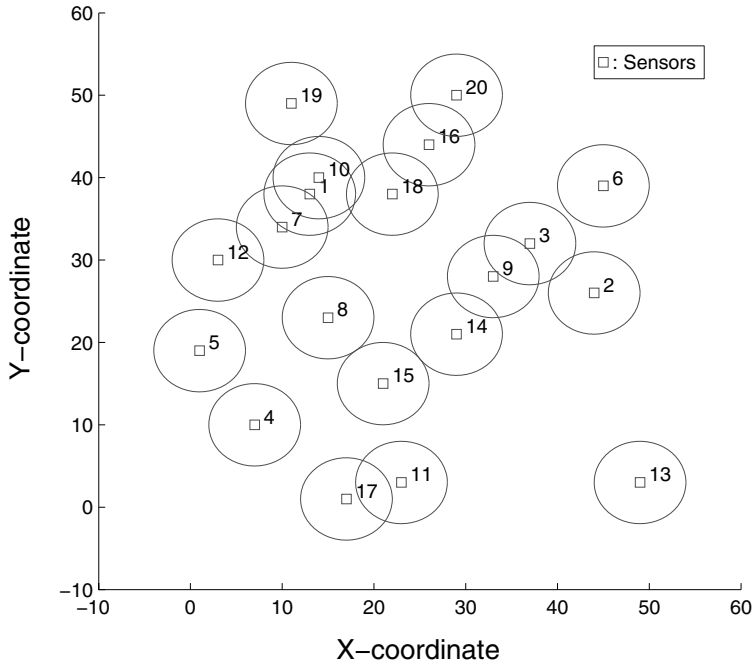

Fig. 2.12. Initial sensor positions after random placement (binary sensor detection model).

Figures 2.12–2.15 present simulation results based on the binary sensor detection model given by Equation (2.1). The initial locations of the sensors are shown in Figure 2.12. Figure 2.13 shows the final sensor positions determined by the VFA algorithm. For the binary sensor detection model, an

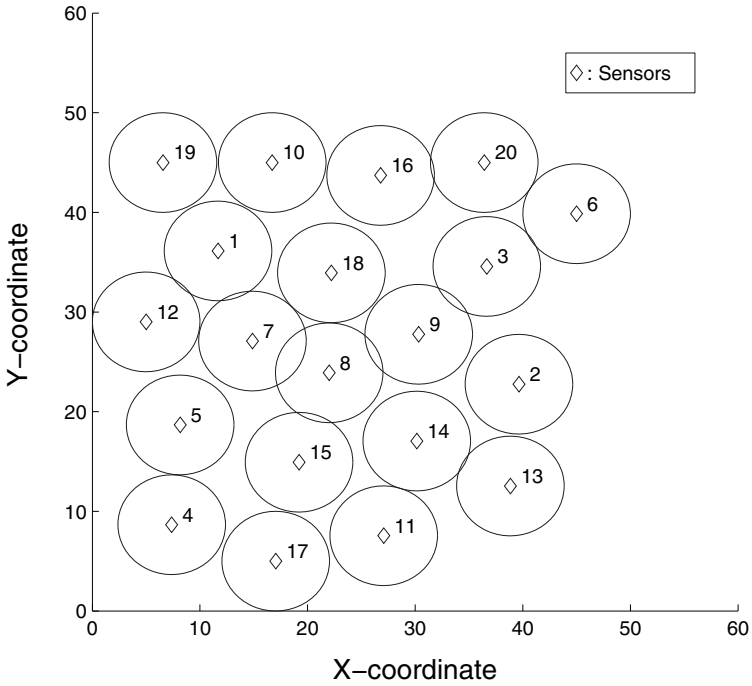

Fig. 2.13. Sensor positions after the execution of the VFA algorithm (binary sensor detection model).

upper bound on the coverage is given by the ratio of the sum of the circle areas (corresponding to sensors) to the total area of the sensor field. For our example, this upper bound evaluates to 0.628 and it is achieved after 28 iterations of the VFA algorithm. Figure 2.14 shows the virtual movement traces of all sensors during the execution of the VFA algorithm. Figure 2.15 shows the improvement in coverage during the execution of the VFA algorithm.

2.3.2 Case Study 2

Figures 2.16–2.18 present simulation results for the probabilistic sensor model given by Equation (2.3). The probabilistic sensor detection model parameters are set as $\lambda = 0.5$, $\beta = 0.5$, and $c_{th} = 0.7$. The initial sensor placements are shown in Figure 2.16. Figure 2.17 shows the final sensor positions determined by the VFA algorithm. Figure 2.18 shows the virtual movement traces of all sensors during the execution of the VFA algorithm. We can see that overlap areas are used to increase the number of grid points whose coverage exceeds the required threshold c_{th}. Figure 2.19 shows the improvement of coverage during the execution of the VFA algorithm. Note that the upper bound for the coverage for the probabilistic sensor detection model in Figure 2.19

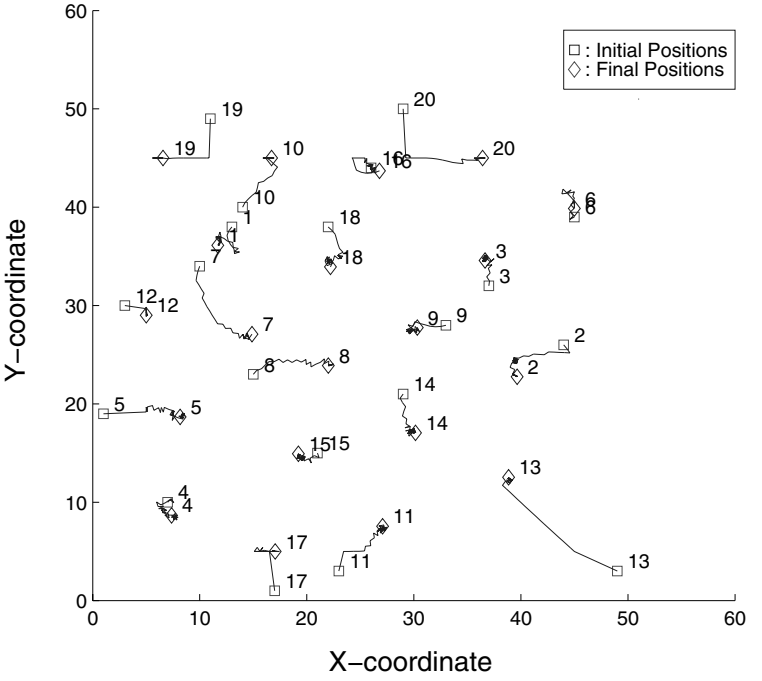

Fig. 2.14. A trace of virtual moves made by the sensors (binary sensor detection model).

(roughly 0.38) is lower than the upper bound for the case of binary sensor detection model in Figure 2.15 (roughly 0.63). This due to the fact that for the simulation results shown here, the coverage for the binary sensor detection model is the fraction of the sensor field covered by the circles. For the probabilistic sensor detection model, even though there are a large number of grid points that are covered, the overall number of grid points with coverage probability greater than the required level is fewer.

2.3.3 Case Study 3

As discussed in Section 2.2, VFA is also applicable to a sensor field containing obstacles and preferential areas. If obstacles are to be avoided, they can be modeled as repulsive force sources in the VFA algorithm. Preferential areas should be covered first, therefore they are modeled as attractive force sources in the VFA algorithm. Figure 2.20–2.23 present simulation results for a 50 by 50 sensor field that contains an obstacle and a preferential area. The binary sensor detection mobel given by Equation (2.1) is used for this simulation. The initial sensor placements are shown in Figure 2.20. Figure 2.21 shows the final sensor positions determined by the VFA algorithm. Figure 2.22 shows

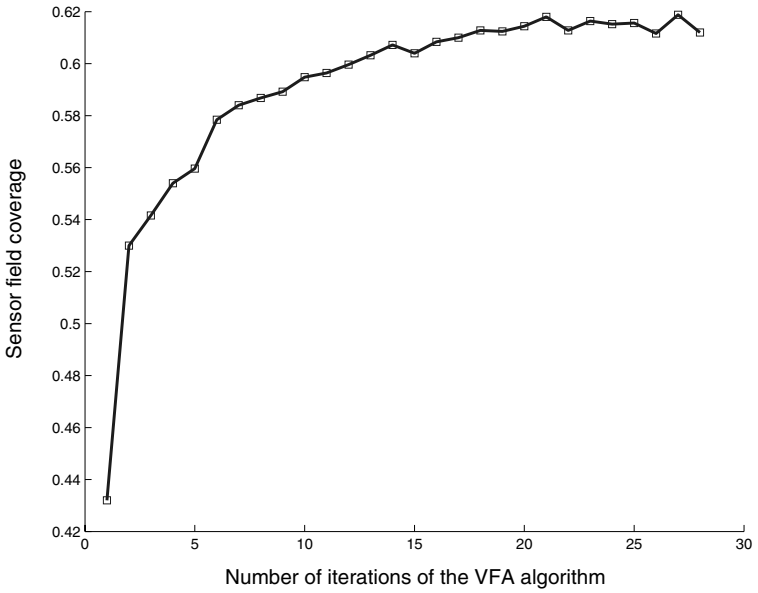

Fig. 2.15. Sensor field coverage improvement by the VFA algorithm (binary sensor detection model).

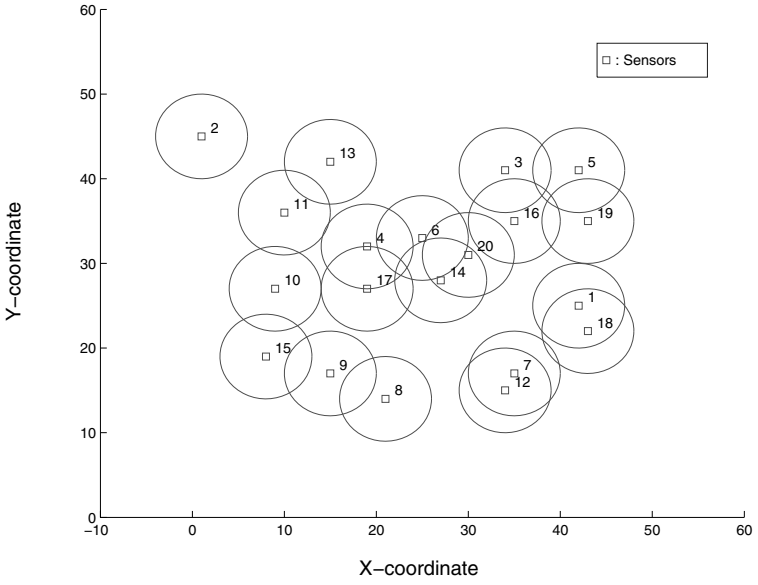

Fig. 2.16. Initial sensor positions after random placement (probabilistic sensor detection model).

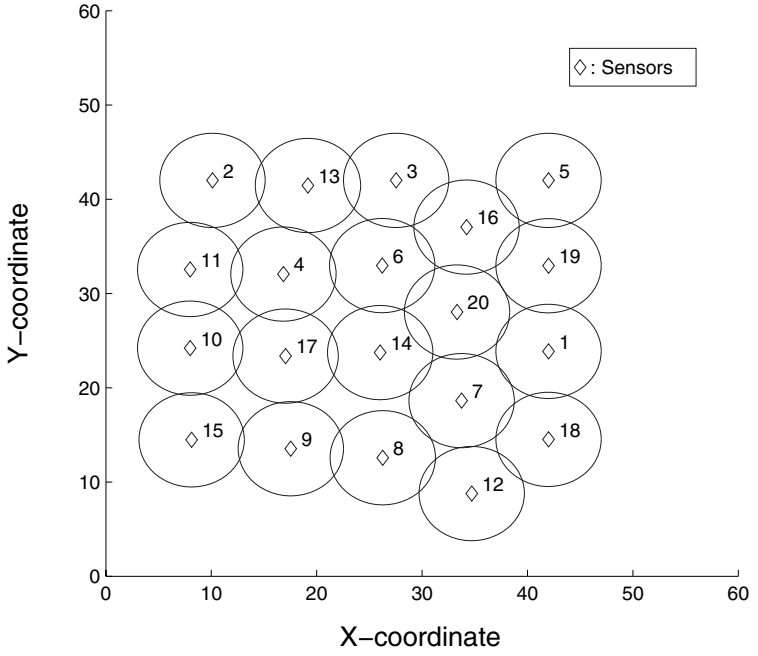

Fig. 2.17. Sensor positions after the execution of the VFA algorithm (probabilistic sensor detection model).

the virtual movement traces of all sensors during the execution of the VFA algorithm. Figure 2.23 shows the improvement of coverage during the execution of the VFA algorithm.

The VFA algorithm does not require much computation time. For Case study 1, the VFA algorithm took only 25 seconds for 30 iterations. For Case study 2, the VFA algorithm took only 3 minutes to complete 50 iterations. Finally for Case study 3, the VFA algorithm took only 48 seconds to complete 50 iterations. Note that these computation times include the time needed for displaying the simulation results on the screen. CPU time is important because sensor redeployment should not take excessive time.

In order to examine how the VFA algorithm scales for larger problem instances, we considered up to 90 sensor nodes in a cluster for a 50 by 50 grid, with $r = 3$, $r_e = 2$, $\lambda = 0.5$ and $\beta = 0.5$ for all cases. For a given number of sensor nodes, we run the VFA algorithm over 10 sets of random deployment results and take the average of the computation time. The results, listed in Table 2.1, show that the CPU time grows slowly with the number of sensors k. For a total of 90 sensors, the CPU time is only 4 minutes on a Pentium III PC. In practice, a cluster head usually has less computational power than a Pentium III PC; however, our results indicate that even if the cluster head

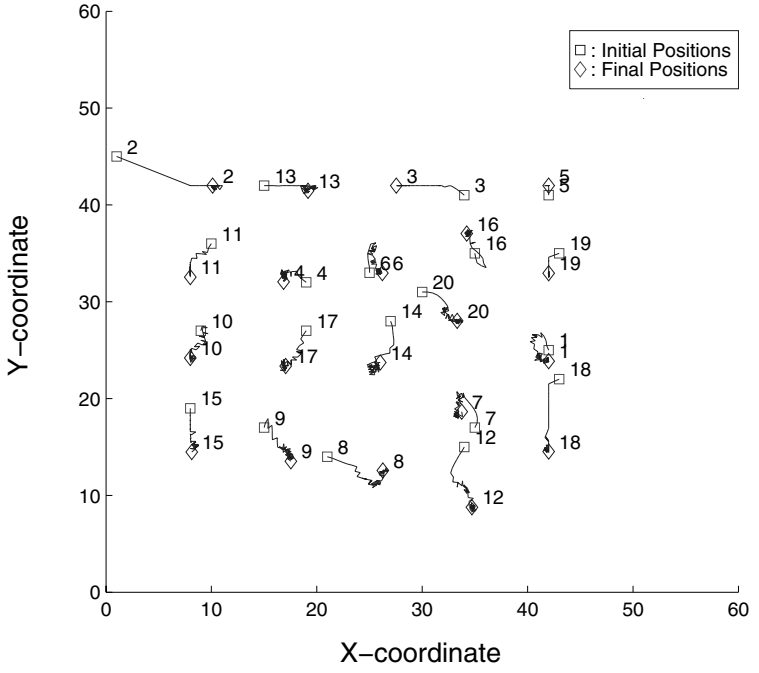

Fig. 2.18. A trace of virtual moves made by the sensors (probabilistic sensor detection model).

has less memory and an on-board processor that runs 10 times slower, the CPU time for the VFA algorithm is reasonable.

Table 2.1. The computation time for the VFA algorithm for larger problem instances.

k	Binary Model	Probabilistic Model	k	Binary Model	Probabilistic Model
40	21 seconds	1.8 minutes	70	46 seconds	3.6 minutes
50	32 seconds	2.2 minutes	80	59 seconds	3.7 minutes
60	38 seconds	3.1 minutes	90	64 seconds	4.0 minutes

2.4 Uncertainty Modeling

The topology of the sensor field, i.e., the locations of the sensors, determines to a large extent the quality and the extent of the coverage provided by the sensor network. However, even if the sensor locations are precomputed for

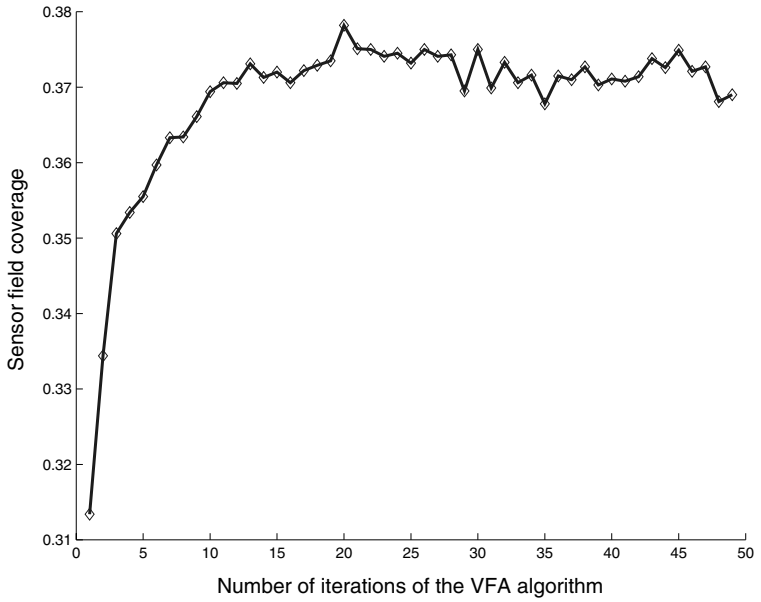

Fig. 2.19. Sensor field coverage achieved using the VFA algorithm (probabilistic sensor detection model).

optimal coverage and resource utilization, there are inherent uncertainties in the sensor locations when the sensors are dispersed, scattered, or airdropped. Thus a key challenge in sensor deployment is to determine an uncertainty-aware sensor field architecture that reduces cost and provides high coverage, even though the exact location of the sensors may not be controllable. We consider the sensor deployment problem in the context of uncertainty in sensor locations subsequent to airdropping[138, 139]. Sensor deployment in such scenarios is inherently non-deterministic and there is a certain degree of randomness associated with the location of a sensor in the sensor field. We present two algorithms for the efficient placement of sensors in a sensor field when the exact locations of the sensors are not known. The proposed approach is aimed at optimizing the number of sensors and determining their placement to support distributed sensor networks. These algorithms are targeted at average coverage as well as at maximizing the coverage of the most vulnerable regions in the sensor field. Experimental results for an example sensor field demonstrate the application of our approach.

In applications such as battlefield surveillance and environmental monitoring, sensors may be dropped from airplanes. Such sensors cannot be expected to fall exactly at predetermined locations; rather there are regions where there is a high probability of sensor being actually located (Figure 2.24). In underwater deployment, sensors may move due to drift or water currents. Furthermore in most real-life situations, it is difficult to pinpoint the exact location

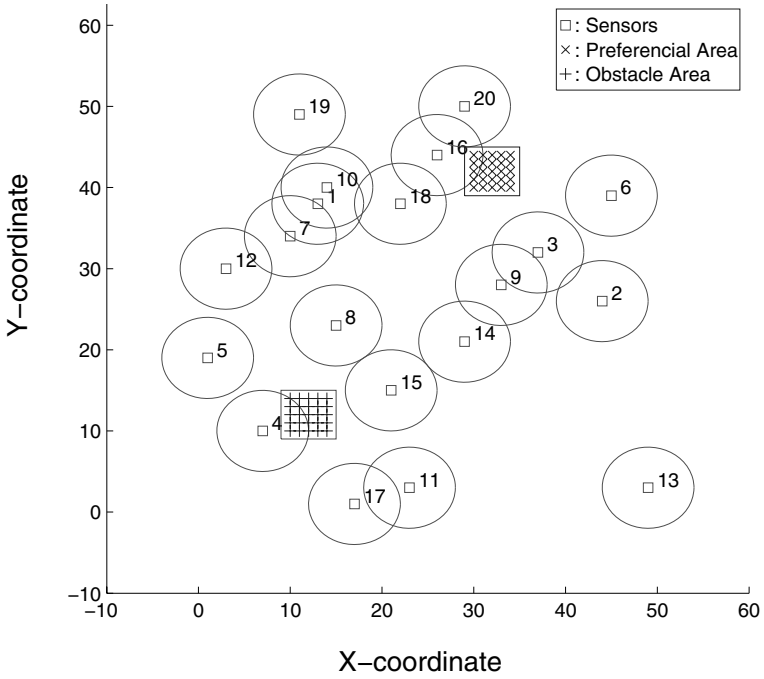

Fig. 2.20. Initial sensor positions after random placement with obstacles and preferred areas.

of each sensor since only a few of the sensors may be aware of their locations. Thus the position of sensors may not exactly known and for every point in the sensor field, there is only a certain probability of a sensor being located at that point.

In this section, we present two algorithms for sensor deployment wherein we assumed that sensor positions are not exactly predetermined. We assume that the sensor locations are calculated before deployment and an attempt is made during the airdrop to place sensors at these locations; however, the sensor placement calculations and coverage optimization are based on a Gaussian model, which assumes that if a sensor is intended for a specific point P in the sensor field, its exact location can be anywhere in a "cloud" surrounding P.

2.4.1 Modeling of Non-Deterministic Placement

During sensor deployment, an attempt is made to place sensors at appropriate predetermined locations by air-dropping or other means. This does not guarantee however that sensors are actually placed at the designated positions, due to unanticipated conditions such as wind, the slope of the terrain, etc. In this case, there is a certain probability of a sensor being located at a particular grid point as a function of the designated location. The deviation about

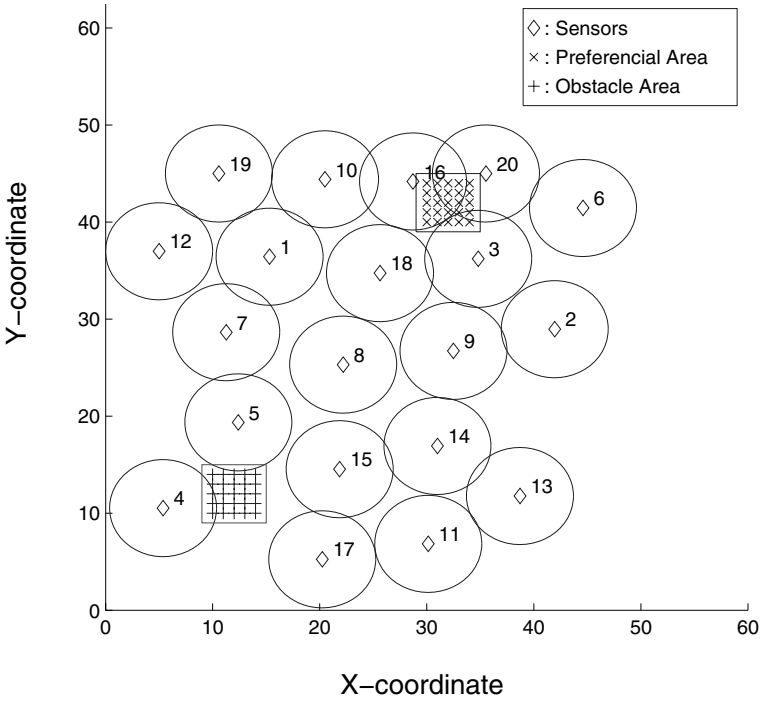

Fig. 2.21. Sensor positions after the execution of the VFA algorithm with obstacles and preferred areas.

the designated sensor locations may be modeled using a Gaussian probability distribution, where the intended coordinates (x, y) serve as the mean values with standard deviation σ_x and σ_y in the x and y dimensions, respectively. Assuming that the deviations in the x and y dimensions are independent, the joint probability density function with mean (x, y) is given by:

$$p_{xy}(x', y') = \frac{e^{-\frac{(x'-x)^2}{2\sigma_x^2} - \frac{(y'-y)^2}{2\sigma_y^2}}}{2\pi\sigma_x\sigma_y}. \tag{2.10}$$

Let us use the notation introduced in the previous section. We still consider a sensor field represented by a $m \times n$ grid, denoted as $Grid$, with S denoting the set of sensor nodes. Let L_S be the set that contains corresponding sensor node locations, i.e. $L_S = \{(x_p, y_p) | s_p \text{ at } (x_p, y_p), s_p \in S\}$. Let A be the total area encompassing all possible sensor locations. To model the uncertainty in sensor locations, the conditional probability $c^*_{ij}(x, y)$ for a grid point (i, j) to be detected by a sensor that is supposed to be deployed at (x, y) is then given by:

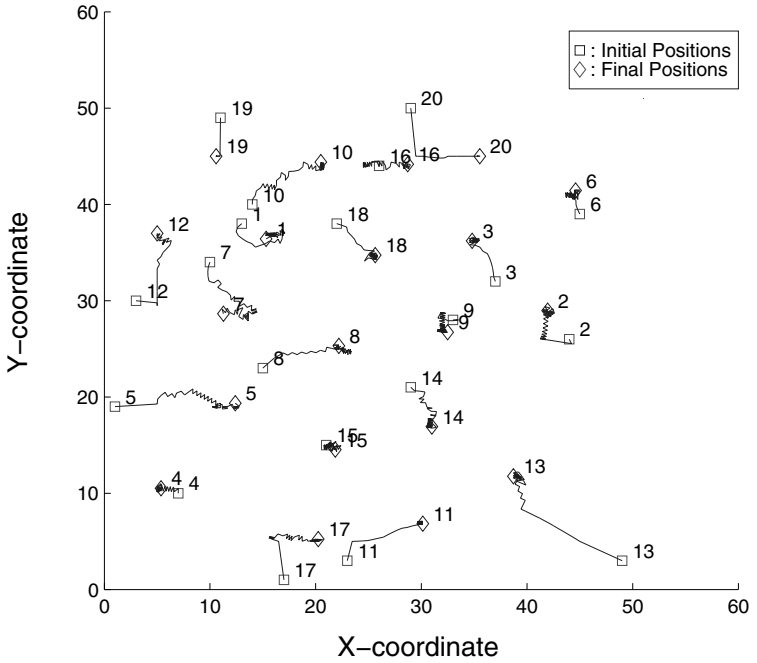

Fig. 2.22. A trace of virtual moves made by the sensors with obstacles and preferred areas.

$$c^*_{ij}(x,y) = \frac{\sum\limits_{(x',y')\in A} c_{ij}(x',y')p_{xy}(x',y')}{\sum\limits_{(x',y')\in A} p_{xy}(x',y')}. \tag{2.11}$$

Based on Equations (2.10) and (2.11), we defined the matrices $C^*_{xy} = [c^*_{ij}(x,y)]_{m\times n}$ and $P = [p_{xy}(x',y')]_A$. Figure 2.25 illustrates the effect of the uncertainty under different variations in the grid point coverage probability $c_{ij}(x,y)$ due to a sample sensor at the designated placement location of $(x,y) = (3,4)$ for a 10 by 10 grid.

2.4.2 Uncertainty-Aware Placement Algorithms

In this section, we introduce the sensor placement algorithm with consideration of uncertainties in sensor locations. The goal of sensor placement algorithms is to determine the minimum number of sensors and their locations such that every grid point is covered with a minimum confidence level. The sensor placement algorithms do not give us the actual location of the sensor but only the mean position of the sensor. It is straightforward to define the miss probability in our sensor deployment scenario. The *miss probability* of a

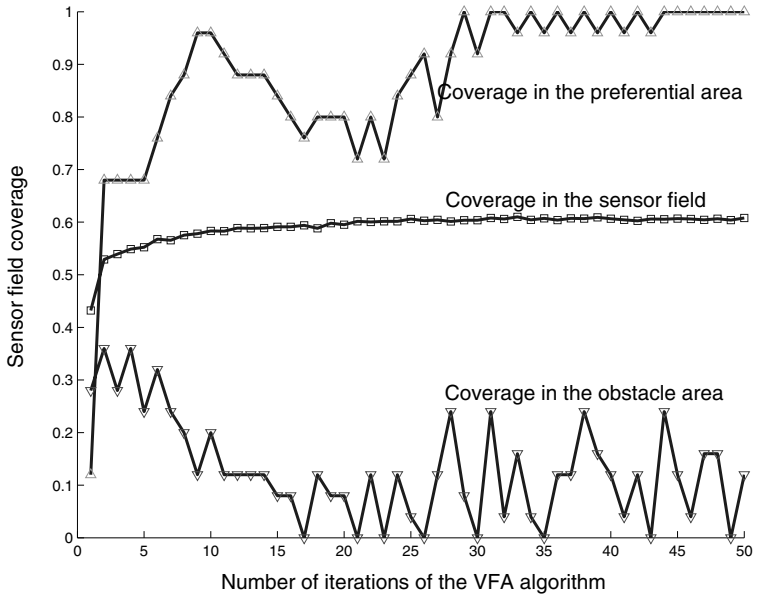

Fig. 2.23. Sensor field coverage achieved using the VFA algorithm with obstacles and preferred areas.

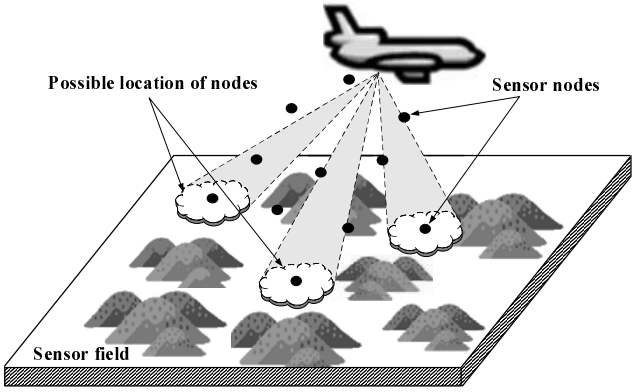

Fig. 2.24. Sensors dropped from airplanes. The clouded region gives the possible region of a sensor location. The black dots within the clouds show the mean (intended) position of a sensor.

grid point (i, j) due to a sensor at (x, y), denoted as $m_{ij}(x, y)$, is given by:

$$m_{ij}(x, y) = 1 - c^*_{ij}(x, y). \tag{2.12}$$

Therefore the miss probability matrix due to a sensor placed at (x, y) is $M_{xy} = [m_{ij}(x, y)]_{m \times n}$. M_{xy} is associated with each grid point and can be pre-determined based on Equations (2.10), (2.11) and (2.12). Since a number

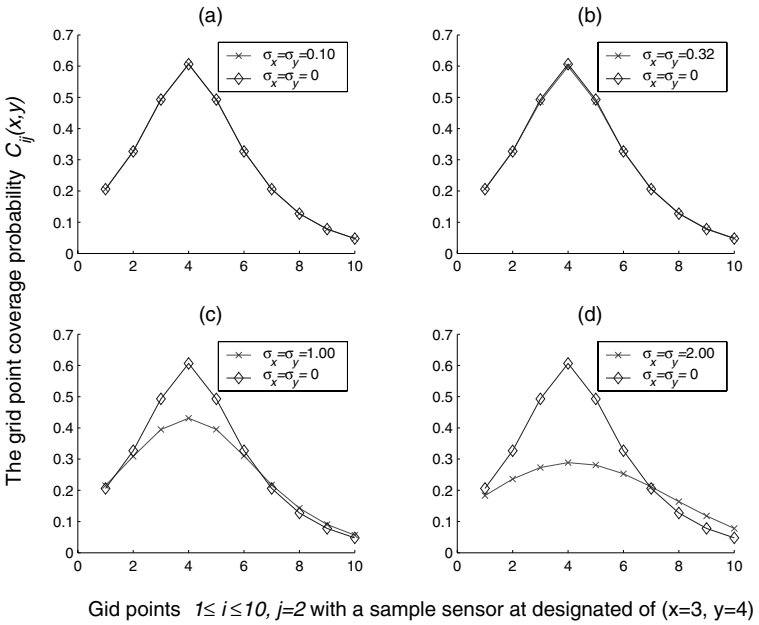

Fig. 2.25. Example of grid point coverage under uncertainty for (a) $\sigma_x = \sigma_y = 0.10$ (b) $\sigma_x = \sigma_y = 0.32$ (c) $\sigma_x = \sigma_y = 1.00$ (d) $\sigma_x = \sigma_y = 2.00$.

of sensors are placed for coverage, we would like to know the miss probability of each grid point due to a set of sensors, namely the collective miss probability. We denote the term *collective miss probability* as m_{ij} and define it in the form of maximum likelihood function as

$$m_{ij} = \prod_{(x,y)\in L_S} m_{ij}(x,y) = \prod_{(x,y)\in L_S} [1 - c^*_{ij}(x,y)]. \tag{2.13}$$

Accordingly we have $M = [m_{ij}]_{m\times n}$ as the collective miss probability matrix over the grid points in the sensor field.

We determine the location of the sensors one at a time. In each step, we find all possible locations that are available on the grid for a sensor, and calculate the overall miss probability associated due to this sensor and those already deployed. We denote the *overall miss probability* due to the newly introduced sensor at grid point (x, y) as $\widetilde{m}(x, y)$, which is defined as

$$\widetilde{m}(x,y) = \sum_{(i,j)\in Grid} m_{ij}(x,y) m_{ij}. \tag{2.14}$$

Based on the $\widetilde{m}(x, y)$ values, where $(x, y) \in Grid$ and $(x, y) \notin L_S$, we can place sensors either at the grid point with the maximum miss probability (the worst coverage case) or the minimum miss probability (the best coverage case).

We refer to the two strategies as MAX_MISS and MIN_MISS, respectively. Therefore, the sensor location can be found based on the following rule. For $(x, y) \in Grid$ and $(x, y) \notin L_S$,

$$\widetilde{m}(x,y) = \begin{cases} \min\{\widetilde{m}(x',y')\}, & \text{if MIN\_MISS is used;} \\ \max\{\widetilde{m}(x',y')\}, & \text{if MAX\_MISS is used.} \end{cases} \tag{2.15}$$

When the best location is found for the current sensor, the collective miss probability matrix M is updated with the newly introduced sensor at location (x, y). This is carried out using Equation (2.16):

$$M = M \cdot M_{xy} = [m_{ij} \cdot m_{ij}(x,y)]_{m \times n}. \tag{2.16}$$

There are two parameters that serve as the termination criterion for the two algorithm. The first is k_{max}, which is the maximum number of sensors that we can afford to deploy. The second is the threshold on the miss probability of each grid point, m_{th}. Our objective is to ensure that every grid point is covered with probability at least $c_{th} = 1 - m_{th}$. Therefore, the rule to stop the further execution of the algorithm is

$$m_{ij} < m_{th} \text{ for all } (i,j) \in Grid \textbf{ or } k > k_{max}, \tag{2.17}$$

where k is the number of deployed sensors. The performance of the proposed algorithm is evaluated using the *average coverage probability* of the grid defined as

$$c_{avg} = \frac{\sum_{(x,y) \in Grid} c_{ij}}{m \cdot n}. \tag{2.18}$$

where c_{ij} is the *collective coverage probability* of a grid point due to all sensors on the grid, defined as

$$\begin{aligned} c_{ij} &= 1 - \prod_{(x,y) \in L_S} m_{ij}(x,y) \\ &= 1 - \{ \prod_{(x,y) \in L_S} [1 - c^*_{ij}(x,y)] \}. \end{aligned} \tag{2.19}$$

We have thus far considered the coverage of only the grid points in the sensor field. In order to provide robust coverage of the sensor field, we also need to ensure that the region that lies between the grid points is adequately covered, i.e., every non-grid point has a miss probability less than the threshold m_{th}. Consider the four grid points in Figure 2.26 that lie on the four corners of a square. Let the distance between these grid points be d^*. The point of intersection of the diagonals of the square is at distance $\frac{d^*}{\sqrt{2}}$ from the four grid points. The following theorem provides a sufficient condition under which the non-grid points are adequately covered by the MIN_MISS and MAX_MISS algorithms.

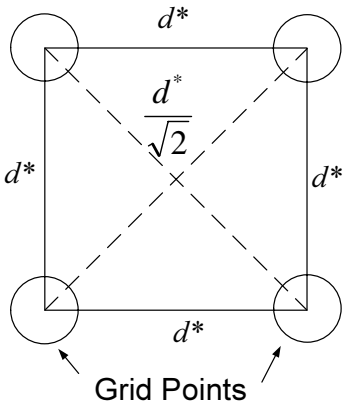

Fig. 2.26. Coverage of non-grid points.

Theorem 2.1. *Let the distance between the grid point P_1 and a potential sensor location P_2 be d. Let the distance between adjacent grid points be d^*. If a value of $d + \frac{d^*}{\sqrt{2}}$ is used to calculate the coverage of grid point P_1 due to a sensor at P_2, and the number of available sensors is adequate, the miss probability of all the non-grid points is less than a given threshold m_{th} when the algorithms MAX_MISS and MIN_MISS terminate.*

Proof: Consider the four grid points in Figure 2.26. The center of square, i.e., the point of intersection of diagonals, is at a distance of $\frac{d^*}{\sqrt{2}}$ from each of the four grid points. Every other non-grid point is at a shorter distance (less than $\frac{d^*}{\sqrt{2}}$) from at least one of the four grid points. Thus if a value of $d + \frac{d^*}{\sqrt{2}}$ is used to determine coverage in the MAX_MISS and MIN_MISS algorithms, we can guarantee that every non-grid point is covered with a probability that exceeds $1 - m_{th}$. ■

In order to illustrate Theorem 1, we consider a 5 by 5 grid with $\alpha = 0.5, \lambda = 0.5, \beta = 0.5$ and $m_{th} = 0.4$. We use Theorem 1 and the MAX_MISS algorithm to determine sensor placement and to calculate the miss probabilities for all the centers of the squares. The results are shown in Figure 2.27 and Figure 2.28 for both sensor detection models. They indicate that the miss probabilities are always less than the threshold m_{th}, thereby ensuring adequate coverage of the non-grid points.

2.4.3 Procedural Description

Note that matrices C_{xy}, M_{xy} and P_A can all be calculated before the actual execution of the placement algorithms. This is illustrated in Figure 2.29 as the pseudocode for the initialization procedure. The initialization procedure is the algorithm overhead which has a complexity of $O((mn)^2)$, where the dimension of the grid is $m \times n$. Once the initialization is done, we may apply either

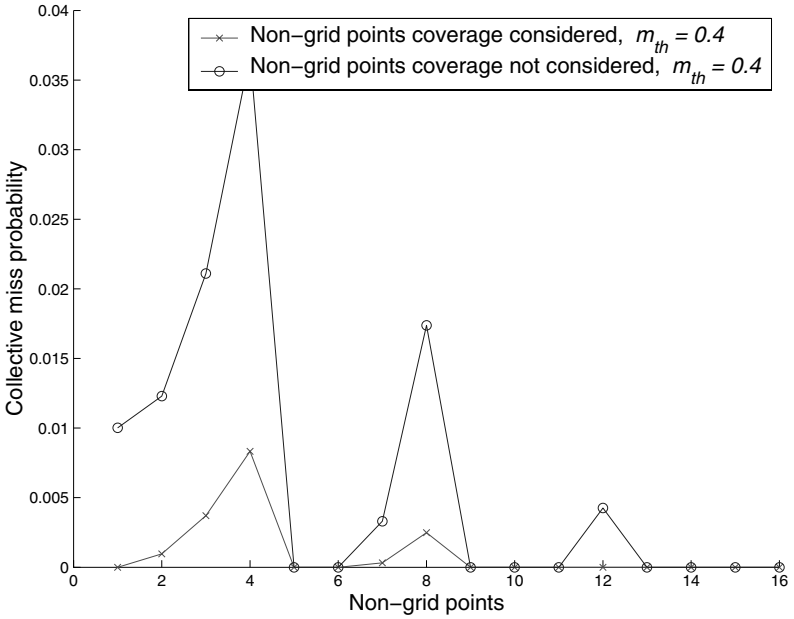

Fig. 2.27. Coverage of non-grid points for the sensor model given by Equation (2.2).

MIN_MISS or MIN_MISS uncertainty-aware sensor placement algorithm using different values for m_{th} and k_{max} with the same C_{xy}, M_{xy} and P_A. Figure 2.30 outlines the main part in pseudocode for the uncertainty-aware sensor placement algorithms. The computational complexity for both MIN_MISS and MAX_MISS is $O(mn)$.

2.5 Simulation Results

Next we present simulation results for the proposed uncertainty-aware sensor placement algorithms MIN_MISS and MAX_MISS using the same testing platform. Note that for practical reasons, we use a truncated Gaussian model because the sensor deviations in a sensor location are unlikely to span the complete sensor field. Therefore $x' - x$ and $y' - y$ in Equation (2.10) are limited to a certain range, which reflects how large the variation is in the sensor locations during the deployment. The maximum error in x direction is denoted as $e^x_{max} = \max(x' - x)$, and the maximum error in y direction is denoted as $e^y_{max} = \max(y' - y)$. We then present our simulation results for different sets of parameters in units of grid point where $m = n = 10$, $\sigma_x = \sigma_y = 0.1, 0.32, 1, 2$, and $e^x_{max} = e^y_{max} = 2, 3, 5$.

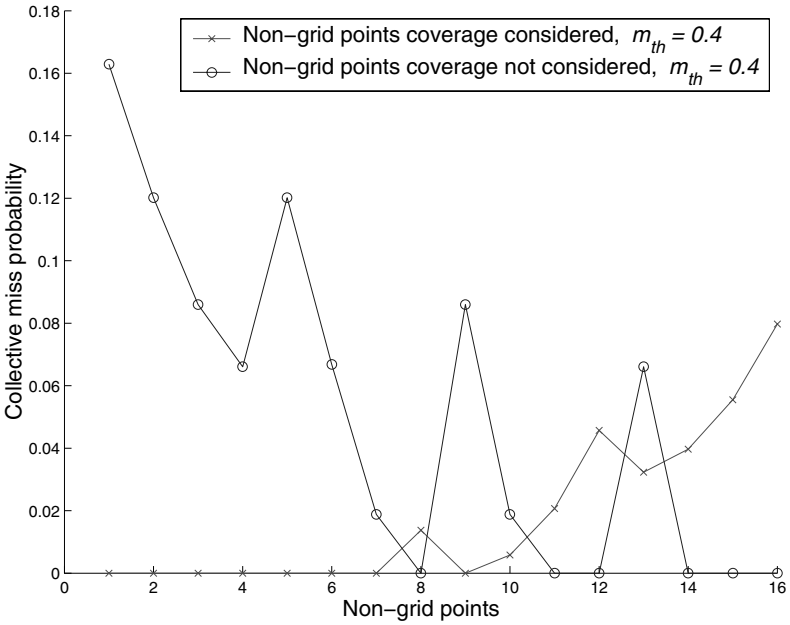

Fig. 2.28. Coverage of non-grid points for the sensor model given by Equation (2.3).

2.5.1 Case Study 1

We first consider the probabilistic sensor detection model given by Equation (2.2) with $\alpha = 0.5$. Figure 2.31 presents the result for the two sensor placement algorithms described by Equation (2.15). Figure 2.32 compares the proposed MIN_MISS and MAX_MISS algorithms with the base case where no location errors are considered, i.e. an uncertainty-oblivious (UO) strategy is followed by setting $\sigma_x = \sigma_y = 0$. We also consider a random deployment of sensors. The results show that MIN_MISS is nearly as efficient as the base uncertainty-oblivious algorithm, yet it is much more robust. Figure 2.33 presents results for the truncated Gaussian models with different maximum errors. Compared to random deployment, MIN_MISS requires more sensors here but we expect random deployment to perform worse in the presence of obstacles. Figure 2.34 for MIN_MISS and MAX_MISS with coverage obtained without location uncertainty. The results show that the MAX_MISS algorithm, which place more sensors for a given coverage threshold, provides higher overall coverage.

2.5.2 Case Study 2

Next, we consider the probabilistic sensor detection model given by Equation (2.3) with $r = 5, r_e = 4, \lambda = 0.5$, and $\beta = 0.5$. Figure 2.35 presents the result for the two sensor placement algorithms described by Equation (2.15). Figure 2.36 compares the proposed MIN_MISS and MAX_MISS algorithms with

Procedure *NDSP_Proc_Init* ($\texttt{Grid}, \sigma_{\texttt{x}}, \sigma_{\texttt{y}}, \alpha, \lambda, \beta$)

01 /* Build the uncertainty area matrix $P = [p_{xy}(x', y')]_A$ */
02 **For** $(x', y') \in A$
03 $\quad p_{xy}(x', y') = \dfrac{e^{-\frac{(x'-x)^2}{2\sigma_x^2} - \frac{(y'-y)^2}{2\sigma_y^2}}}{2\pi\sigma_x\sigma_y}$
04 **End**
05 /* Build the miss probability matrix for all grid points. */
06 **For** grid point $(x, y) \in Grid$
07 $\quad$ /* Build C_{xy}, C^*_{xy}, and M_{xy} for sensor node at (x, y). */
08 $\quad$ **For** grid point $(i, j) \in Grid$
09 $\quad\quad$ /* Non-grid points coverage are considered based on Theorem 1. */
09 $\quad\quad$ $d_{ij}(x, y) = \sqrt{(x-i)^2 + (y-j)^2} + \frac{d^*}{\sqrt{2}}$;
10 $\quad\quad$ /* Calculate the grid point coverage probability based on the sensor detection model. */
11 $\quad\quad$ Calculate $c_{ij}(x, y)$:
12 $\quad\quad$ /* Sensor detection model 1. */
13 $\quad\quad$ Model 1: $c_{ij}(x, y) = e^{-\alpha d_{ij}(x,y)}$
14 $\quad\quad$ /* Sensor detection model 2. */
15 $\quad\quad$ Model 2: $c_{ij}(x, y) = \begin{cases} 0, & \text{if } r + r_e \le d_{ij}(x, y) \\ \mathrm{e}^{-\lambda a^\beta}, & \text{if } |r - d_{ij}(x, y)| < r_e \\ 1, & \text{if } r - r_e \ge d_{ij}(x, y) \end{cases}$
16 $\quad\quad$ /* Modeling of uncertainty in sensor node locations. */
17 $\quad\quad$ $c^*_{ij}(x, y) = \dfrac{\sum_{(x',y') \in A} c_{ij}(x', y') p_{xy}(x', y')}{\sum_{(x',y') \in A} p_{xy}(x', y')}$;
18 $\quad\quad$ /* The miss probability matrix */
19 $\quad\quad$ $m_{ij}(x, y) = 1 - c^*_{ij}(x, y)$;
20 $\quad$ **End**
21 $\quad$ /* Use the obstacle mask matrix based on the a priori knowledge about the terrain. */
22 $\quad$ **If** Obstacles exist
23 $\quad\quad$ $C_{xy} = C_{xy} \cdot ObstacleMaskMatrix$
24 $\quad\quad$ Revise M_{xy}.
25 $\quad$ **End**
26 **End**
27 /* Initially the overall miss probability matrix is set to I. */
28 $M = [m_{ij}]_{m \times n} = [1]_{m \times n}$;

Fig. 2.29. Initialization pseudocode.

the base case where no location errors are considered. Figure 2.37 presents results for the truncated Gaussian models with different maximum errors.

Procedure *NDSP_Proc_Main* (type, k_{max}, m_{th}, Grid, C_{xy}, M_{xy}, P_A, M)

01 /* Initially no sensors have been placed yet. */
02 Set $S = \phi$; $L_S = \{\phi\}$; $k = |S|$;
03 /* Repeatedly placing sensors until requirement is satisfied.*/
04 **Rep eat**
05 /* Evaluate the miss probability due to a sensor at (x, y). */
06 **For** grid point $(x, y) \in Grid$ **And** $(x, y) \notin L_S$
07 Retrieve $M_{xy} = [m_{ij}(x, y)]_{m \times n}$

$$= [1 - c^*_{ij}(x, y)]_{m \times n}$$

$$= \left[1 - \frac{\sum_{(x',y') \in A} c_{ij}(x', y') p_{xy}(x', y')}{\sum_{(x',y') \in A} p_{xy}(x', y')} \right]_{m \times n}$$

08 /* Miss probability if sensor node is placed at (x, y) */
09 $\widetilde{m}(x, y) = \sum_{(i,j) \in Grid} m_{ij}(x, y) m_{ij}$;
10 **End**
11 /* Place sensor node using selected algorithm. */
12 **If** *type* =MIN_MISS
13 Find $(x, y) \in Grid$ and $(x, y) \notin L_s$ such that
$\widetilde{m}(x, y) = \min\{\widetilde{m}(x', y')\}$, $(x', y') \in Grid$;
14 **Else** /* MAX_MISS */
15 Find $(x, y) \in Grid$ and $(x, y) \notin L_s$ such that
$\widetilde{m}(x, y) = \max\{\widetilde{m}(x', y')\}$, $(x', y') \in Grid$;
16 **End**
17 /* Save the information of sensor node just placed. */
18 Set $k = k + 1$;
19 Set $L_S = L_S \cup \{(x, y)\}$;
20 Set $S = S \cup \{s_k\}$;
20 /* Update current overall miss probability matrix. */
21 **For** grid point $(i, j) \in Grid$
22 $m_{ij} = m_{ij} \cdot m_{ij}(x, y)$;
23 **End**
24 /* Check if the placement requirement is satisfied. */
25 **Until** $m_{ij} < m_{th}$ for all $(i, j) \in Grid$ **Or** $k > k_{max}$;

Fig. 2.30. Pseudocode for sensor placement algorithm.

Figure 2.38 compares the coverage based on Equation (2.18) for MIN_MISS and MAX_MISS with coverage obtained without location uncertainty. We notice that due to the different probability values as a reflection of the confidence level in sensor responses from these two different models, the results in sensor placement are also different. Compared with Case study 1, this sensor detec-

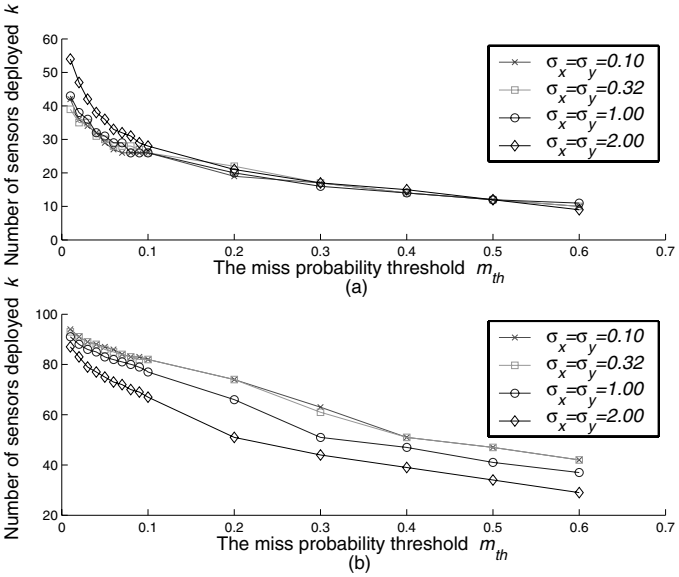

Fig. 2.31. Number of sensors required as a function of the miss probability threshold with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5$, for (a) MIN_MISS (b) MAX_MISS.

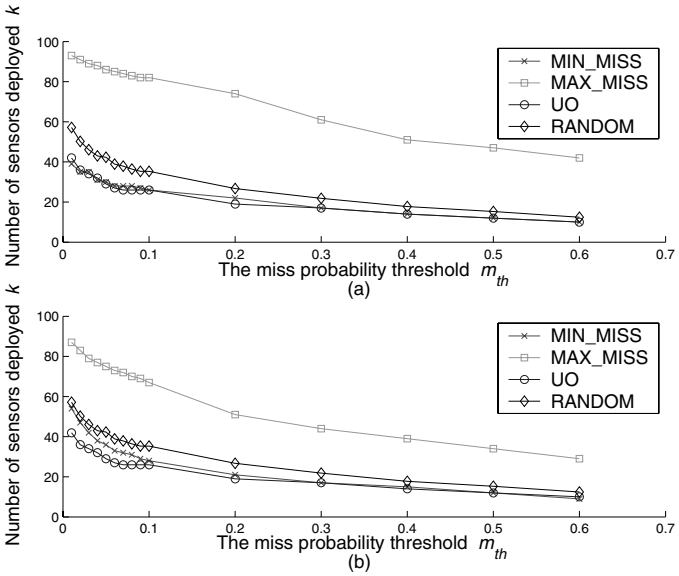

Fig. 2.32. Number of sensors required for various placement schemes with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5$, and (a) $\sigma_x = \sigma_y = 0.32$ (b) $\sigma_x = \sigma_y = 2$.

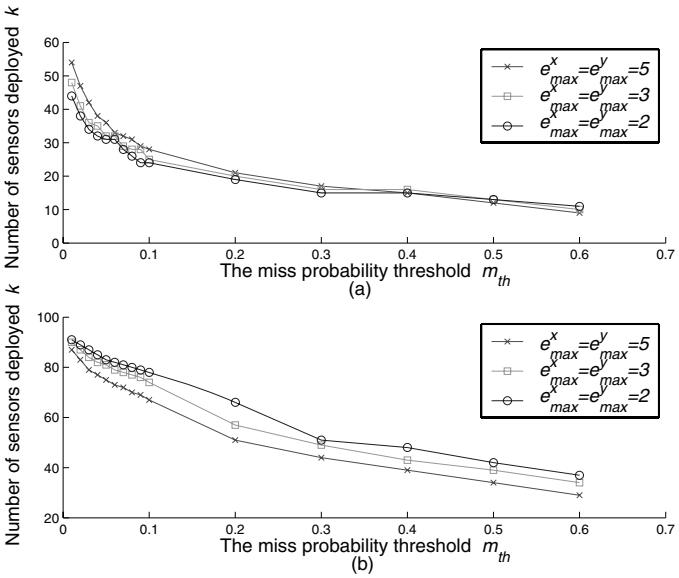

Fig. 2.33. Comparisons in different truncated Gaussian models with $\alpha = 0.5, \sigma_x = \sigma_y = 2$ for (a) MIN_MISS (b) MAX_MISS.

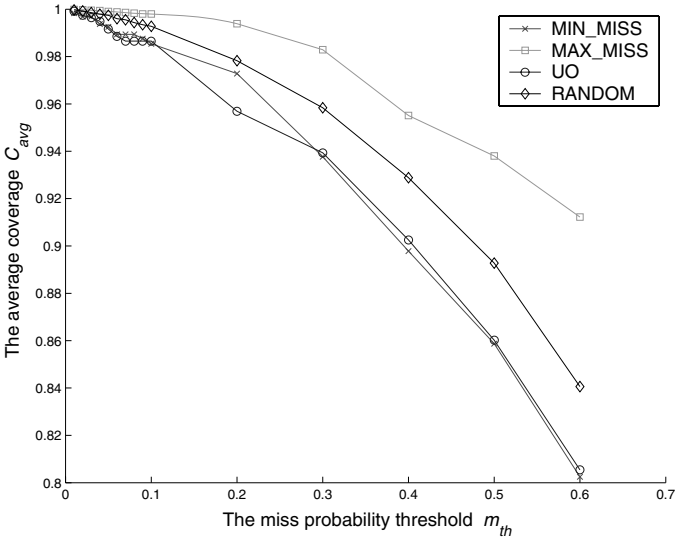

Fig. 2.34. Comparison in average coverage for various placement schemes with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5, \sigma_x = \sigma_y = 0.32$.

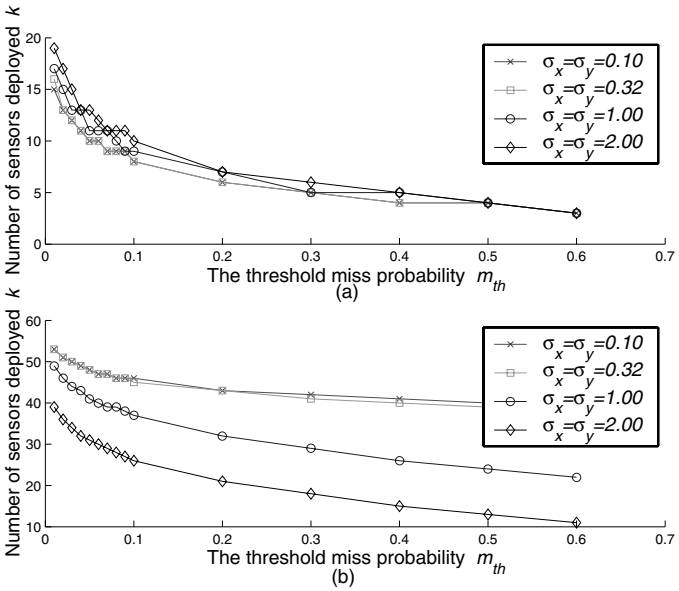

Fig. 2.35. Number of sensors required as a function of the miss probability threshold with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5$, for (a) MIN_MISS (b) MAX_MISS.

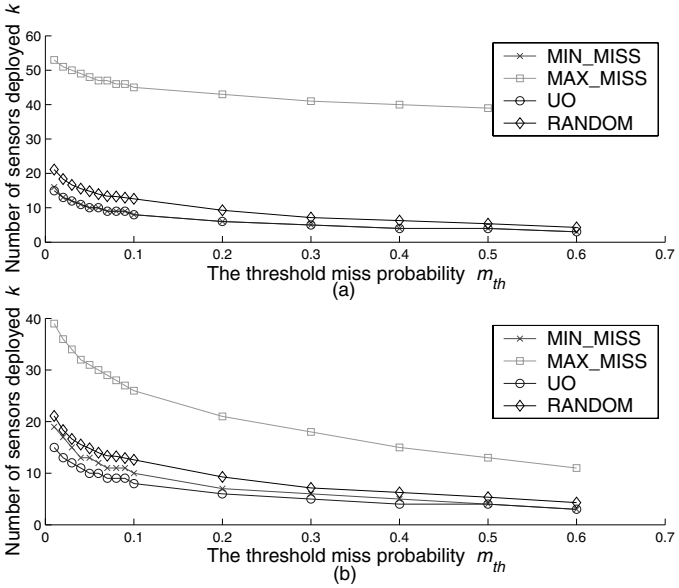

Fig. 2.36. Number of sensors required for various placement schemes with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5$, and (a) $\sigma_x = \sigma_y = 0.32$ (b) $\sigma_x = \sigma_y = 2$.

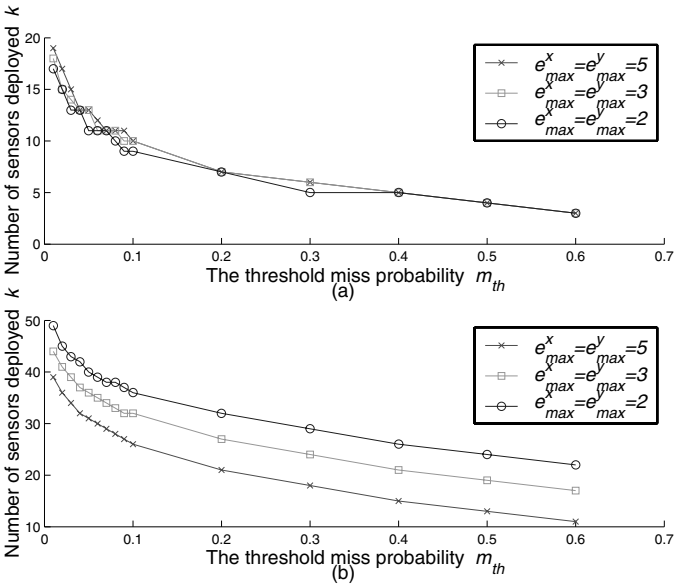

Fig. 2.37. Comparisons in different truncated Gaussian models with $\alpha = 0.5, \sigma_x = \sigma_y = 2$ for (a) MIN_MISS (b) MAX_MISS.

tion model with the selected model parameters as $\lambda = 0.5$ and $\beta = 0.5$ requires less number of sensor nodes for the same miss probability threshold. Part of the reason is due to the fact that in Equation (2.3), we have full confidence in sensor responses for grid points that are very close to the sensor node, i.e. $c_{ij}(x, y) = 1$ if $r - r_e \geq d_{ij}(x, y)$. However, this case study shows that the proposed sensor deployment algorithms do not depend on any specific type of sensor models. The sensor detection model can be viewed as a plug-in module when different types of sensors are encountered in applying the deployment algorithms.

2.5.3 Case Study 3

Next we consider a terrain model with the existence of obstacles. We have manually placed one obstacle that occupies grid points $(7, 3), (7, 4)$, and another obstacle that occupies grid points $(3, 5), (4, 5), (5, 5)$. They are marked as "Obstacle" in Figure 2.3, which gives the layout of the setup for this case study. We have evaluated the proposed algorithms on the sensor detection model in Case study 2, which is given by Equation (2.3) with the same model parameters as $r = 5, r_e = 4, \lambda = 0.5$, and $\beta = 0.5$. Figure 2.39 presents results for the truncated Gaussian models with different maximum errors. Figure 2.40 compares the coverage based on Equation (2.18) for MIN_MISS and MAX_MISS with coverage obtained without location uncertainty. It is obvious that because of the existence of obstacles, the actual range of sensor

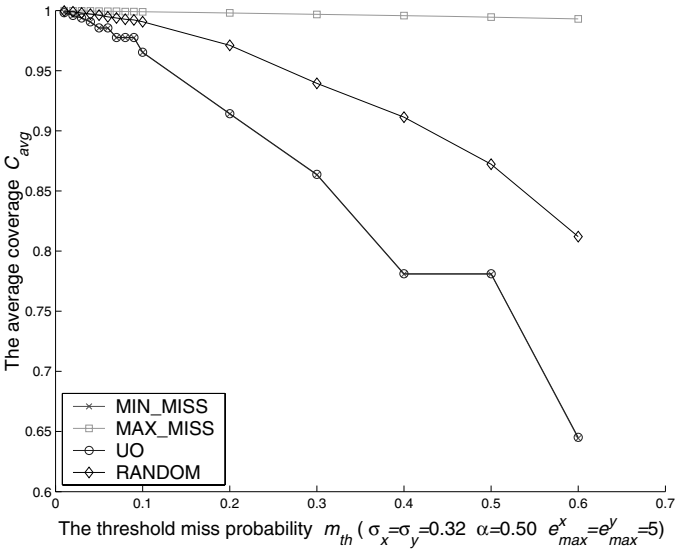

Fig. 2.38. Comparison in average coverage for various placement schemes with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5, \sigma_x = \sigma_y = 0.32$.

detection due to the line-of-sight principle. Therefore, the reduction in sensor detection range causes an increase in the number of sensors required for the same miss probability threshold, as shown in Figure 2.39 and Figure 2.40.

2.6 Discussion

In this chapter, we have discussed two important aspects in sensor node deployment for wireless sensor networks. The proposed VFA algorithm introduced in Section 2.2 improves the sensor field coverage considerably compared to random sensor placement. The sensor placement strategy is centralized at the cluster level since every cluster head makes redeployment decisions for the nodes in its cluster. Nevertheless, the clusters make deployment decisions independently, hence there is a considerable degree of decentralization in the overall sensor deployment. The virtual force in the VFA algorithm is calculated with a grid point being the location indicator and the distance between two grid points being a measure of distance. Furthermore, in our simulations, the preferential areas and the obstacles are both modeled as rectangles. The VFA algorithm however is also applicable for alternative location indicators, distance measures, and models of preferential areas and obstacles. Hence the VFA algorithm can be easily extended to heterogeneous sensors, where sensors may differ from each other in their detection modalities and parameters.

In Section 2.4, we have formulated an optimization problem on uncertainty-aware sensor placement. A minimum number of sensors are deployed to pro-

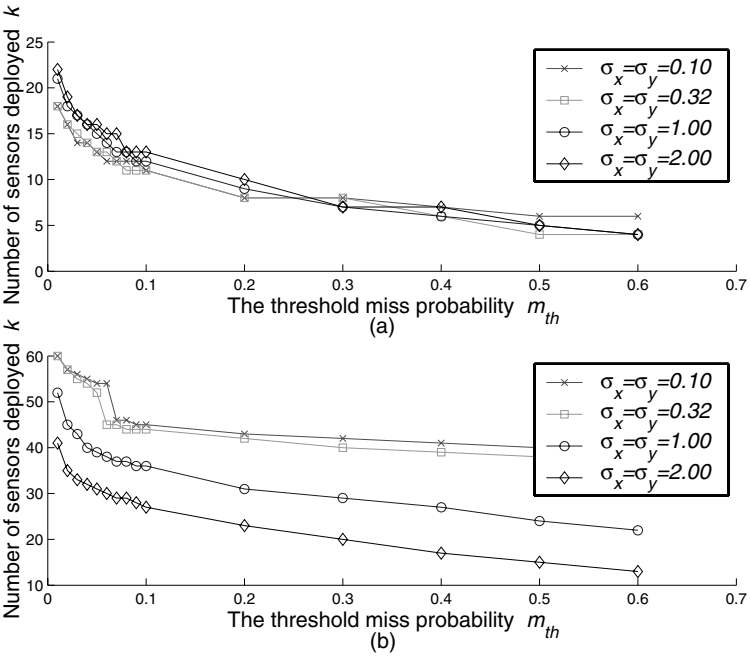

Fig. 2.39. Number of sensors required as a function of the miss probability threshold in presence of obstacles with $\alpha = 0.5, e^x_{max} = e^y_{max} = 5$ for (a) MIN_MISS (b) MAX_MISS.

vide sufficient grid coverage of the sensor field though the exact sensor locations are not known. The sensor location has been modeled as a random variable with a Gaussian probability distribution. We have presented two polynomial-time algorithms to optimize the number of sensors and determine their placement in an uncertainty-aware manner. The proposed algorithms address coverage optimization under constraints of imprecise detections and terrain properties.

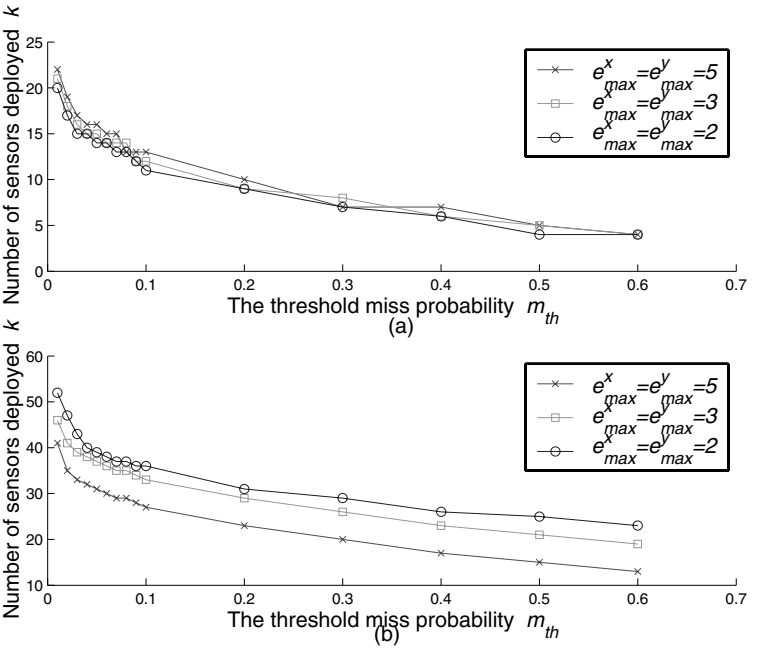

Fig. 2.40. Comparisons in different truncated Gaussian models in presence of obstacles with $\alpha = 0.5, \sigma_x = \sigma_y = 2$ for (a) MIN_MISS (b) MAX_MISS.

http://www.springer.com/978-1-85233-951-7

Scalable Infrastructure for Distributed Sensor Networks

Iyengar, S.S.

2005, XIV, 194 p., Hardcover
ISBN: 978-1-85233-951-7