Sensor Deployment and Target Localization in Distributed Sensor Networks

YI ZOU and KRISHNENDU CHAKRABARTY
Duke University

The effectiveness of cluster-based distributed sensor networks depends to a large extent on the coverage provided by the sensor deployment. We propose a virtual force algorithm (VFA) as a sensor deployment strategy to enhance the coverage after an initial random placement of sensors. For a given number of sensors, the VFA algorithm attempts to maximize the sensor field coverage. A judicious combination of attractive and repulsive forces is used to determine the new sensor locations that improve the coverage. Once the effective sensor positions are identified, a one-time movement with energy consideration incorporated is carried out, that is, the sensors are redeployed, to these positions. We also propose a novel probabilistic target localization algorithm that is executed by the cluster head. The localization results are used by the cluster head to query only a few sensors (out of those that report the presence of a target) for more detailed information. Simulation results are presented to demonstrate the effectiveness of the proposed approach.

Categories and Subject Descriptors: C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*distributed networks*; *wireless communication*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and Embedded Systems

General Terms: Algorithms, Performance, Management

Additional Key Words and Phrases: Cluster-based sensor networks, cluster head, sensor field coverage, sensor placement, virtual force

1. INTRODUCTION

Distributed sensor networks (DSNs) are important for a number of strategic applications such as coordinated target detection, surveillance, and localization. The effectiveness of DSNs 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 paper, we focus on sensor placement strategies that maximize the coverage for a given number of sensors within a cluster in cluster-based DSNs.

As an initial 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,

Authors' address: Department of Electrical and Computer Engineering, Hudson Hall, PO Box 90291, Duke University, Durham, NC 27708; email: {yz1,krish}@ee.duke.edu.

where DSNs 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 parts of the sensor field. The key idea of this paper is that 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 is based on disk packing theory [Locateli and Raber 2002] and the virtual force field concept from robotics [Howard et al. 2002]. 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.

We also propose a novel target localization approach based on a two-step communication protocol between the cluster head and the sensors within the cluster. Since the energy consumption in DSNs increases significantly during periods of activity, which may be triggered, for example, by a moving target [Bhardwaj and Chandrakasan 2002], we propose an energy-conserving method for target localization in cluster-based DSNs. In the first step, sensors detecting a target report the event to the cluster head. The amount of information transmitted to the cluster head is limited; in order to save power and bandwidth, the sensor only reports the presence of a target, and it does not transmit detailed information such as signal strength, confidence level in the detection, imagery or time series data. Based on the information received from the sensor and the knowledge of the sensor deployment within the cluster, the cluster head executes a probabilistic scoring-based localization algorithm to determine likely position of the target. The cluster head subsequently queries a subset of sensors that are in the vicinity of these likely target positions.

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, that is, 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 a realistic probabilistic model in which the probability that the sensor detects a target depends on the relative position of the target within the circle. The details of the probabilistic model are presented in Section 1.

The organization of the paper is as follows. In Section 2, we review prior research on topics related to sensor deployment in DSNs. In Section 3, we present details of the VFA algorithm. In Section 4, we present the target localization

algorithm that is executed by the cluster head. In Section 5, we present simulation results using the proposed sensor deployment strategy for various situations. Section 6 presents conclusions and outlines directions for future work.

2. RELATED PRIOR WORK

Sensor deployment problems have been studied in a variety of contexts [Brooks and Iyengar 1997; Iyengar et al. 1995; Qi et al. 2001; Varshney 1996]. In the area of adaptive beacon placement and spatial localization, a number of techniques have been proposed for both fine-grained and coarse-grained localization [Bulusu et al. 2001; Heidemann and Bulusu 2001].

Sensor deployment and sensor planning for military applications are described in Musman et al. [1997], where a general sensor model is used to detect elusive targets in the battlefield. The sensor model is characterized by a window, which includes physical sensor model parameters, sensor location, terrain characteristics, and the data collected in a certain period of time. The sensor coverage analysis is based on a hypothesis of possible target movements and sensor attributes. This analysis generates all possible routes of targets movements. Bayesian networks are used to calculate the probability that a certain target is detected in a particular area during particular time intervals. However, the proposed DSNs framework in Musman et al. [1997] requires a great deal of *a priori* knowledge about possible targets. Hence, it is not applicable in scenarios where there is no information about potential targets in the environment.

The deployment of sensors for coverage of the sensor field has been considered for multi-robot exploration [Howard et al. 2002]. Each robot can be viewed as a sensor node in such systems. An incremental deployment algorithm is used in which sensor nodes are deployed one by one in an adaptive fashion. Each new deployment of a sensor is based on the sensed information from sensors deployed earlier. The first sensor is placed randomly. A drawback of this approach is that it is computationally expensive. As the number of sensors increases, each new deployment results in a relatively large amount of computation.

The problem of evaluating the coverage provided by a given placement of sensors is discussed in Meguerdichian et al. [2001]. The major concern here is the self-localization of sensor nodes; sensor nodes are considered to be highly mobile and they move frequently. An optimal polynomial-time algorithm that uses graph theory and computational geometry constructs is used to determine the best-case and the worst-case coverage.

Radar and sonar coverage also present several related challenges [Priyantha et al. 2000]. Radar and sonar netting optimization are of great importance for detection and tracking in a surveillance area. Based on the measured radar cross-sections and the coverage diagrams for the different radars, the authors in Priyantha et al. [2000] propose a method for optimally locating the radars to achieve satisfactory surveillance with limited radar resources.

Sensor placement on two- and three-dimensional grids has been formulated as a combinatorial optimization problem, and solved using integer linear programming in Chakrabarty et al. [2001, 2002]. This approach suffers from two main drawbacks. First, computational complexity makes the approach

infeasible for large problem instances. Second, the grid coverage approach relies on "perfect" sensor detection, that is, a sensor is expected to yield a binary yes/no detection outcome in every case. However, because of the inherent uncertainty associated with sensor readings, sensor detection must be modeled probabilistically [Dhillon et al. 2002].

It is well known, however, that there is inherent uncertainty associated with sensor readings; hence, sensor detections must be modeled probabilistically [Dhillon et al. 2002]. A probabilistic optimization framework for minimizing the number of sensors for a two-dimensional grid has been proposed recently [Dhillon et al. 2002]. This algorithm attempts to maximize the average coverage of the grid points. Finally, there exists a close resemblance between the sensor placement problem and the art gallery problem (AGP) addressed by the art gallery theorem [O'Rourke 1987]. The AGP problem can be informally stated as that of determining the minimum number of guards required to cover the interior of an art gallery. (The interior of the art gallery is represented by a polygon.) The AGP has been solved optimally in two dimension and shown to be NP-hard in the three-dimensional case. Several variants of AGP have been studied in the literature, including mobile guards, exterior visibility, and polygons with holes. Other related work includes the placement of a given number of sensors to reduce communication cost [Kasetkasem and Varshney 2001] and optimal sensor placement for a given target distribution [Penny 1998].

Our proposed algorithm differs from prior methods in several ways. First, we consider both the binary sensor detection model and probabilistic detection model to handle sensors with both high and low detection accuracy. Second, the amount of computation is limited since we perform a one-time computation and sensor locations are determined at the same time for all the sensor nodes. Third, our approach improves upon an initial random placement, which offers a practical sensor deployment solution. Finally, we investigate the relationship between sensor placement within a cluster and target localization by the cluster head in an effort to conserve energy whenever there are activities in the DSN.

3. VIRTUAL FORCE ALGORITHM

In this section, we describe the underlying assumptions and the virtual force algorithm (VFA).

3.1 Preliminaries

For a cluster-based sensor network architecture, we make the following assumptions:

—After the initial random deployment, all sensor nodes are able to communicate with the cluster head. This communication is necessary only for the transmission of the new locations to the nodes. This is done only once per node and does not require large amount of data to be transferred; therefore, the energy consumed for this purpose is ignored.

—The cluster head is responsible for executing the VFA algorithm and managing the one-time movement of sensors to the desired locations

—In order to minimize the network traffic and conserve energy, sensors only send a yes/no notification message to the cluster head when a target is detected. The cluster head intelligently queries a subset of sensors to gather more detailed target information.

The VFA algorithm combines the ideas of potential field [Howard et al. 2002] and disk packing [Locateli and Raber 2002]. 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 are placed too close to each other, the "closeness" being measured by a predetermined 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 predetermined threshold is used here), they exert positive forces on each other. This ensures that a globally uniform sensor placement is achieved.

Consider an n by m sensor field grid and assume that there are k sensors deployed in the random deployment stage. Each sensor has a detection range r. Assume sensor s_i is deployed at point (x_i, y_i). For any point P at (x, y), we denote the Euclidean distance between s_i and P as $d(s_i, P)$, that is, $d(s_i, P) = \sqrt{(x_i - x)^2 + (y_i - y)^2}$. Equation (1) shows the binary sensor model [Chakrabarty et al. 2001, 2002] that expresses the coverage $c_{xy}(s_i)$ of a grid point P by sensor s_i.

$$c_{xy}(s_i) = \begin{cases} 1, & \text{if } d(s_i, P) < r \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The binary sensor model assumes that sensor readings have no associated uncertainty. In reality, sensor detections are imprecise, hence the coverage $c_{xy}(s_i)$ needs to be expressed in probabilistic terms. In this work, we assume the sensor model given by Equation (2), which is motivated in part by Elfes [1990]. Our approach can also be used with alternative sensor models that are based on radio signal propagation models in which signal strength decays as a power of the distance [Rappaport 1996]; the sensor placement and localization algorithms are independent of the sensor models.

$$c_{xy}(s_i) = \begin{cases} 0, & \text{if } r + r_e \le d(s_i, P) \\ e^{-\lambda a^{\beta}}, & \text{if } r - r_e < d(s_i, P) < r + r_e \\ 1, & \text{if } r - r_e \ge d(s_i, P) \end{cases} \tag{2}$$

where $r_e (r_e < r)$ is a measure of the uncertainty in sensor detection, $a = d(s_i, P) - (r - r_e)$, 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 1. Note that distances are measured in units of grid points. Figure 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

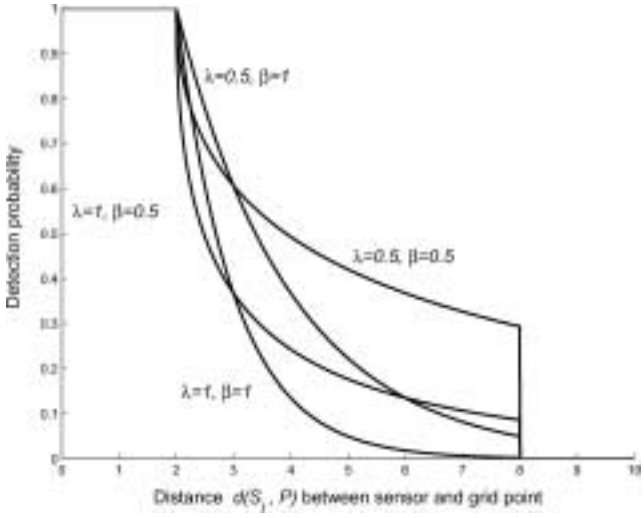

Fig. 1. Probabilistic sensor detection model.

different detection probabilities, which can be viewed as the characteristics of various types of physical sensors.

3.2 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 the total force action on sensor s_i be denoted by $\vec{F}_i$. Note that $\vec{F}_i$ is a vector whose orientation is determined by the vector sum of all the forces acting on s_i. Let the force exerted on s_i by another sensor s_j be denoted by $\vec{F}_{ij}$.

In addition to the positive and negative forces due to other sensors, a sensor s_i 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, for example, obstacles such as building and trees in the line of sight for infrared sensors, uneven surface, elevations for hilly terrain, and so on. 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

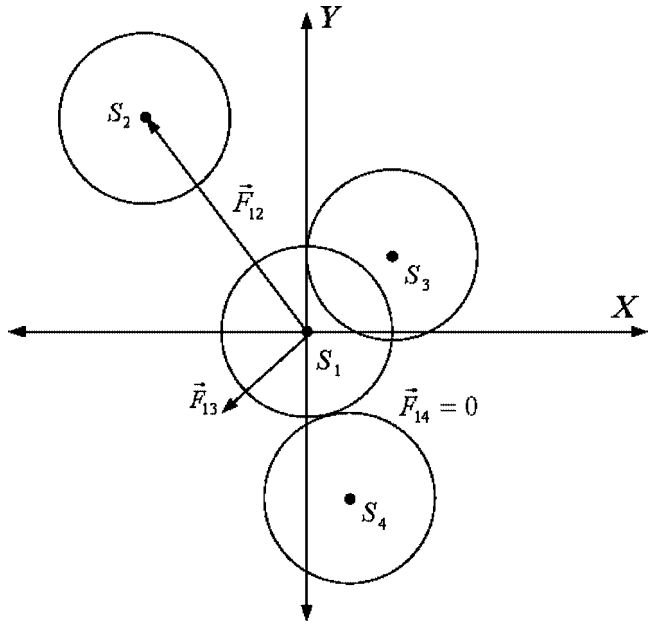

Fig. 2. An example of virtual forces with four sensors.

appropriately. In this work, we let $\vec{F}_{iA}$ be the total (attractive) force on s_i due to preferential coverage areas, and let $\vec{F}_{iR}$ be the total (repulsive) force on s_i due to obstacles. The total force $\vec{F}_i$ on s_i can now be expressed as,

$$\vec{F}_i = \sum_{j=1,\ j\neq i}^{k} \vec{F}_{ij} + \vec{F}_{iR} + \vec{F}_{iA}. \tag{3}$$

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

$$\vec{F}_{ij} = \begin{cases} (w_A(d_{ij} - d_{th}), \alpha_{ij}) & \text{if } d_{ij} > d_{th} \\ 0, & \text{if } d_{ij} = d_{th} \\ (w_R \frac{1}{d_{ij}}, \alpha_{ij} + \pi), & \text{if otherwise} \end{cases} \tag{4}$$

where d_{ij} is the Euclidean distance between sensor s_i and s_j, d_{th} is the threshold on the distance between s_i and s_j, α_{ij} is the orientation (angle) of a line segment from s_i to s_j, 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. The force $\vec{F}_1$ on s_1 is given by $\vec{F}_1 = \vec{F}_{12} + \vec{F}_{13} + \vec{F}_{14}$. 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. Note that d_{th} is a predetermined 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.

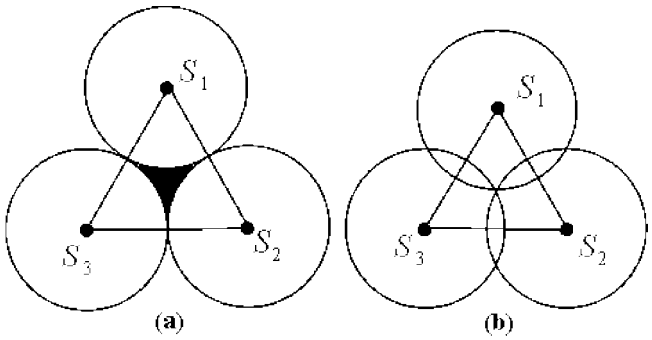

Fig. 3. Nonoverlapped and overlapped sensor coverage areas.

If $r_e \approx 0$ and we use the binary sensor detection model given by Equation (1), we attempt to make d_{ij} 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 3(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 3(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) 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 (x, y) lying in the overlap region of sensors s_i and s_j. Let $c_{xy}(s_i, s_j)$ 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

$$c_{x,y}(s_i, s_j) = 1 - (1 - c_{x,y}(s_i))(1 - c_{x,y}(s_j)) \tag{5}$$

where $c_{xy}(s_i)$ and $c_{xy}(s_j)$ were defined in Section 3.1. Since the term $(1 - c_{x,y}(s_i))(1 - c_{x,y}(s_j))$ expresses the probability that neither s_i nor s_j covers grid point at (x, y), the probability that the grid point (x, y) is covered is given by Equation (5). Let c_{th} be the desired coverage threshold for all grid points. This implies that

$$\min_{x,y}\{c_{x,y}(s_i, s_j)\} \geq c_{th}. \tag{6}$$

Note that Equation (5) 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

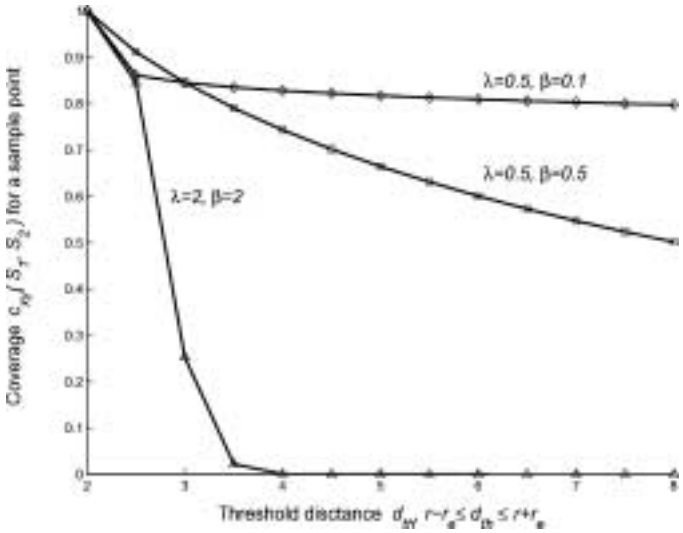

Fig. 4. Coverage versus d_{th} of a sample point inside the overlapped area of s_1 and s_2.

in this case is given by

$$c_{x,y}(S_{ov}) = 1 - \prod_{s_i \in S_{ov}} (1 - c_{x,y}(s_i)). \tag{7}$$

As shown in Equation (4), 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 $c_{xy}(s_i) > c_{th}$. Let us consider the three sensors s_1, s_2, and s_3 in Figure 3(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 4–6 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}.

3.3 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 $d_{max}(s_i)$ to denote the maximum distance that sensor s_i can move in the repositioning phase. To simplify the discussion without loss of generality, we assume $d_{max}(s_i) = d_{max}(s_j) = d_{max}$, $i, j = 1, 2, \ldots, k$. During the execution of the VFA algorithm, for each sensor, whenever the distance from the current

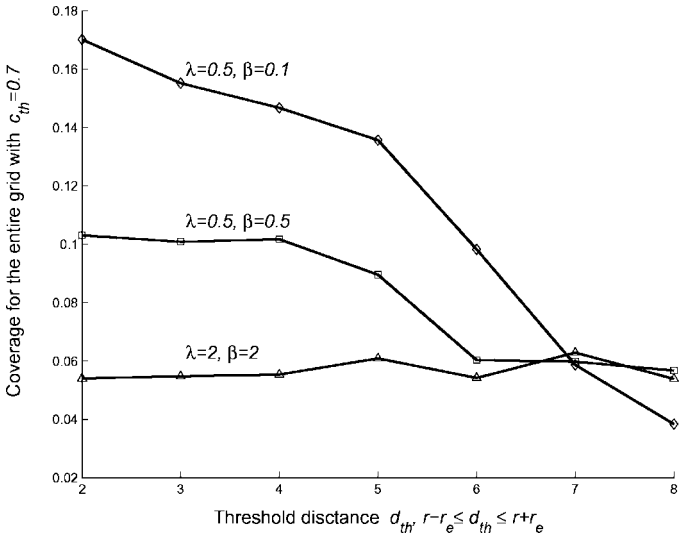

Fig. 5. Coverage versus d_{th} with $c_{th} = 0.7$ and different λ and β.

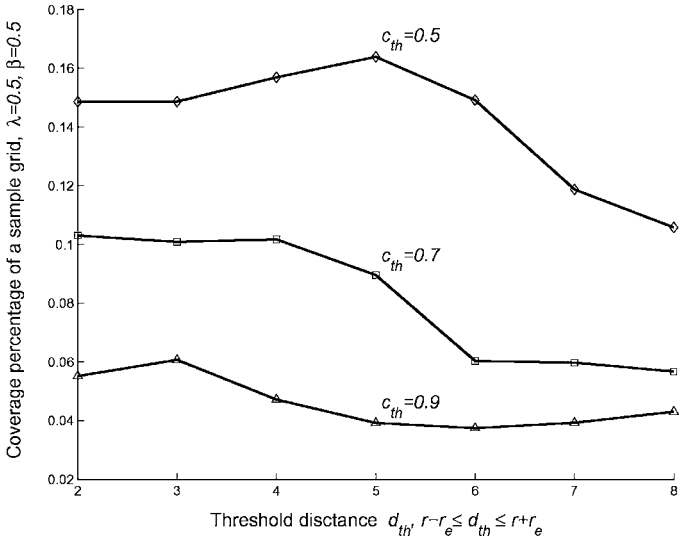

Fig. 6. Coverage vs. d_{th} with $\lambda = 0.5$ and $\beta = 0.5$ and $c_{th} = 0.5$, 0.7 and 0.9.

virtual position to the initial position reaches the distance limit d_{max}, any virtual forces on this sensor are disabled. For sensor s_i, let $(x, y)_{rand}$ be the initial location obtained from the random deployment, and $(x, y)_{virtual}$ be the location generated by the VFA algorithm. The energy constraint can be described as:

$$\vec{F}_i = \begin{cases} 0, & \text{if } d((x, y)_{rand}, (x, y)_{virtual}) \geq d_{max} \\ \vec{F}_i, & \text{otherwise (i.e., the force is unchanged).} \end{cases} \quad (8)$$

VFA Data Structures: Grid, $\{s_1, s_2, \ldots, s_k\}$

/* n_P is the number of preferential area blocks (attractive forces) and n_O is the number of obstacle blocks (repulsive forces). S_{xy}, k_{xy} and p_table_{xy} are used for localization. $(x, y)_{VFA}$ is the final position found by the VFA algorithm. d_{max} is the energy constraint on the sensor repositioning 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_{xy}: $c_{xy}(\{s_1, s_2, \ldots, s_k\})$, S_{xy}, k_{xy}, p_table_{xy};
6 Sensor s_i structure:
$(x, y)_{rand}$, $(x, y)_{virtual}$, $(x, y)_{VFA}$, i, r, r_e, α, β, d_{max};

Fig. 7. Data structures used in the VFA algorithm.

Therefore, the virtual force $\vec{F}_i$ given by Equation (3) on sensor s_i 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 (8), 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.

3.4 Procedural Description of the VFA Algorithm

Figure 7 shows the data structure of the VFA algorithm, and Figure 8 shows the implementation details in pseudocode form. For an 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 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, for example, if the sensors are initially placed along the circumference of a circle such that all virtual forces are balanced. Since these specific scenarios are extremely unlikely for random deployment, they are not considered in this paper. We will extend the VFA algorithm to handle such boundary cases in future work.

Procedure *Virtual_Force_Algorithm* (Grid, $\{s_1, s_2, \ldots, s_k\}$)

1 Set $loops = 0$;
2 Set $MaxLoops =$ **MAX_LOOPS**;
3 **While** ($loops < MaxLoops$)
4 /* coverage evaluation */
5 **For** $P(x, y)$ in Grid, $x \in [1, width]$, $y \in [1, height]$
6 **For** $s_i \in \{s_1, s_2, \ldots, s_k\}$
7 Calculate $c_{xy}(s_i, P)$ from the sensor model using $(d(s_i, P), c_{th}, d_{th}, \alpha, \beta)$;
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** $s_i \in \{s_1, s_2, \ldots, s_k\}$
15 Calculate $\vec{F}_{ij}$ using $d(s_i, s_j)$, d_{th}, w_A, w_R;
16 Calculate $\vec{F}_{iA}$ using $d(s_i, PA_1, \ldots, PA_{n_P})$, d_{th};
17 Calculate $\vec{F}_{iR}$ using $d(s_i, OA_1, \ldots, OA_{n_O})$, d_{th};
18 $\vec{F}_i = \sum \vec{F}_{ij} + \vec{F}_{iR} + \vec{F}_{iA}$, $j \in [1, k]$, $j \neq i$;
19 **End**
20 /*move sensors virtually */
21 **For** $s_i \in \{s_1, s_2, \ldots, s_k\}$
22 /* energy constraint on the sensor movement */
23 **If** $d((x, y)_{rand}, (x, y)_{virtual}) \geq d_{max}$
24 Set $\vec{F}_i = 0$;
25 **End**
26 $\vec{F}_i$ virtually moves s_i to its next position;
27 **End**
28 Set $loops = loops + 1$; /* continue to next iteration */
29 **End**

Fig. 8. Pseudocode of the VFA algorithm.

4. TARGET LOCALIZATION

In our two-step communication protocol, when a sensor detects a target, it sends an event notification to the cluster head. In order to conserve power and bandwidth, the message from the sensor to the cluster head is kept very small; in fact, the presence or absence of a target can be encoded in just one bit. Detailed information such as detection strength level, imagery, and time series data are stored in the local memory and provided to the cluster head upon subsequent queries. Based on the information received from the sensors within the cluster, the cluster head executes a probabilistic localization algorithm to determine candidate target locations, and it then queries the sensor(s) in the vicinity of the target.

4.1 Detection Probability Table

After the VFA algorithm is used to determine the final sensor locations, the cluster head generates a detection probability table for each grid point. The detection probability table contains entries for all possible detection reports from those sensors that can detect a target at this grid point. Let us assume that a grid point $P(x, y)$ is covered by a set of k_{xy} sensors, denoted as S_{xy}, $|S_{xy}| = k_{xy}$,

Table I. Example Probability Table

i	$d_1d_2d_3$	$p\_table_{xy}(i),\ 0 \le i < 2^{k_{xy}},\ k_{xy} = 3$
0	000	$(1 - 0.5736) \times (1 - 1) \times (1 - 0.5736) = 0.0$
1	001	$(1 - 0.5736) \times (1 - 1) \times 0.5736 = 0.0$
2	010	$(1 - 0.5736) \times 1 \times (1 - 0.5736) = 0.1819$
3	011	$(1 - 0.5736) \times 1 \times 0.5736 = 0.2446$
4	100	$0.5736 \times (1 - 1) \times (1 - 0.5736) = 0.0$
5	101	$(1 - 0.5736) \times (1 - 1) \times 0.5736 = 0.0$
6	110	$0.5736 \times 1 \times (1 - 0.5736) = 0.2446$
7	111	$0.5736 \times 1 \times 0.5736 = 0.3290$

$0 \le k_{xy} \le k$, and $S_{xy} \subseteq \{s_1, s_2, \ldots, s_k\}$. The probability table is built on the power set of S_{xy} since there are $2^{k_{xy}}$ possibilities for k_{xy} sensors in reporting an event. These $2^{k_{xy}}$ cases include the event that none of the sensors detect anything (represented by the binary string as "00...0") as well as the event that all of the sensors (represented by the binary string as "11...1"). Thus the probability table for grid point (x, y) then contains $2^{k_{xy}}$ entries, defined as,

$$p\_table_{xy}(i) = \prod_{s_j \in S_{xy}} p_{xy}(s_j, i) \tag{9}$$

where $0 \le i \le 2^{k_{xy}}$, and $p_{xy}(s_j, i) = c_{x,y}(s_j)$ if s_j detects a target at grid point $P(x, y)$; otherwise $p_{xy}(s_j, i) = 1 - c_{x,y}(s_j)$. Table I gives an example of the probability tables on a 5 by 5 grid with three sensors deployed.

Consider the grid point (2, 4) in Figure 9 which is covered by all three sensors s_1, s_2, and s_3 with probabilities as 0.57, 1 and 0.57, respectively. For the three sensors s_1, s_2, and s_3, there are a total of eight possibilities for their combined event detection at grid point (2, 4). For example, the binary string 110 denotes the possibility that s_1 and s_2 report a target but s_3 does not report a target. For each such possibility $d_1d_2d_3$ ($d_1, d_2, d_3 \in \{0, 1\}$) for a grid point, we calculate the conditional probabilities that the cluster head receives $d_1d_2d_3$ given that a target is present at that grid point. For our example, these conditional probabilities are listed in Table I. Consider the binary string 110, the conditional probability associated with this possibility is given by $p\_table_{24}(6) = p_{24}(s_1, 6)p_{24}(s_2, 6)p_{24}(s_3, 6) = 0.57 \times 1 \times (1 - 0.57) = 0.24$. Note that the probability table generation is only a one-time cost. Once the probability table is generated, there is no need to refresh it unless sensor locations are changed.

4.2 Score-Based Ranking

After the probability table is generated for all the grid points, localization is done by the cluster head if a target is detected by one or more sensors. We use an inference method based on the established probability table. When at time instant t, the cluster head receives positive event message from $k(t)$ sensors, it uses the grid point probability table to determine which of these sensors are most suitable to be queried for more detailed information. Detailed target reporting consumes more energy consumption and it needs more bandwidth.

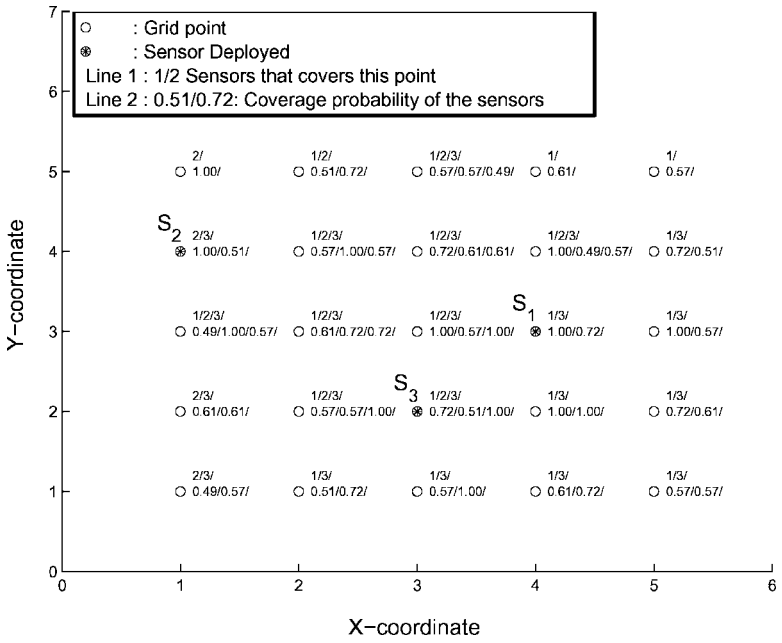

Fig. 9. Example of grid point probability table.

Therefore, the cluster head cannot afford to query all the sensors for detailed reports. There is also an inherent redundancy in sensor detection information so it is not necessary to query all sensors. Our scoring approach is able to select the most suitable sensors for this purpose.

Consider the 10 by 10 grid shown in Figure 10. There are five sensors deployed, $k = 5$, $r = 2$, and $r_e = 1$. The zigzag-shaped line is the target movement trace. The target starts to move at $t = t_{start}$ from the grid point marked as "Start" and finishes at $t = t_{end}$ at the grid point marked as "End." Figure 11 gives the score report at the time instant t_{start} when the target is present at "Start."

Assume $S_{rep}(t)$ is the set of sensors that have reported the detection of an object at time t, $S_{rep,xy}(t)$ is the set of sensors that can detect a target at point $P(x, y)$ and have also reported the detection of an object at time t. Obviously, $S_{rep,xy}(t) \subseteq S_{rep}(t)$ and $S_{rep,xy}(t) \subseteq S_{xy}$ since $S_{rep,xy}(t) = S_{rep}(t) \cap S_{xy}$. The score of the grid point $P(x, y)$ at time instant t is calculated as follows:

$$SCORE_{xy}(t) = p\_table_{xy}(i(t)) \times w_{xy}(t) \tag{10}$$

where $i(t)$ is the index of the p_table_{xy} at time t. The parameter $i(t)$ is calculated from S_{xy} and $S_{rep,xy}$. The parameter $p\_table_{xy}(i(t))$ corresponds to the conditional probability that the cluster head receives this event information given that there was a target at $P(x, y)$. The weight $w_{xy}(t)$ reflects the confidence level in this reporting event for this particular grid point. In our previous work [Zou and Chakrabarty 2003], we have used the weight factor $w_{xy}(t) = \frac{k_{rep,\,xy}(t)}{k_{rep}(t)}$; this is sufficient for selecting sensors in order to conserve energy. However, in order to

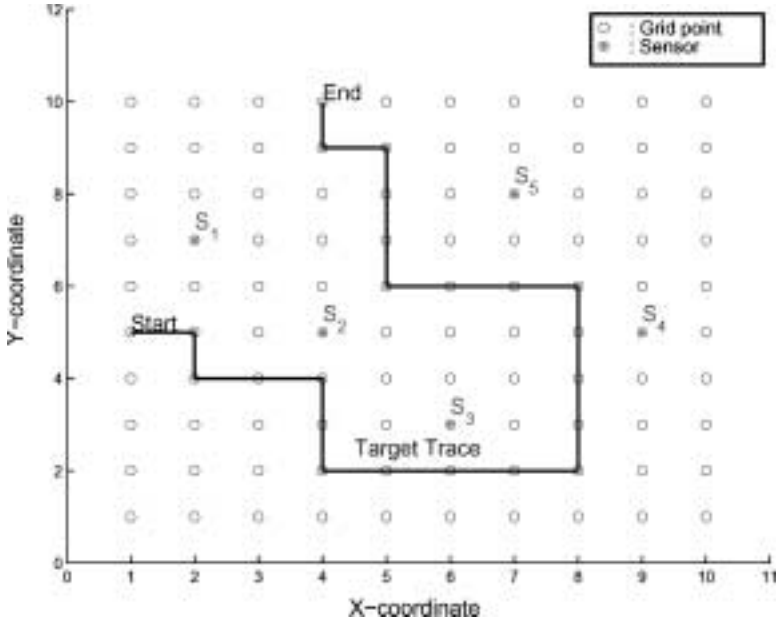

Fig. 10. Example sensor field with a moving target.

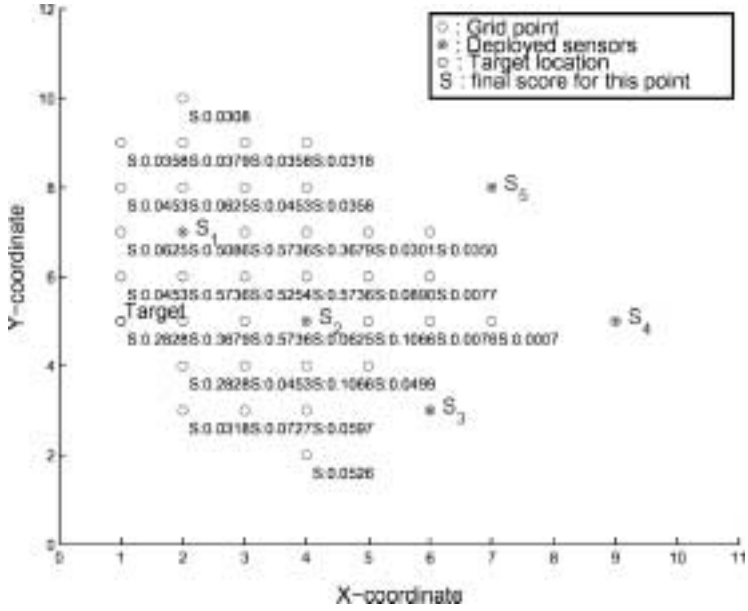

Fig. 11. Scoring results for target in the example sensor field at t_{start}. s_1 and s_2 have reported.

Table II. Scoring Calculation Example for t at t_{start}

(x, y)	S_{xy}	$S_{rep,xy}(t)$	$w_{xy}(t)$	$p\_table_{xy}(i(t))$	$SCORE_{xy}(t)$
...	...	...	...	...	...
(1, 6)	s_1	s_1	0.25	0.7248	0.0453
(2, 6)	s_1, s_2	s_1, s_2	1.00	0.5736	0.5736
(3, 6)	s_1, s_2	s_1, s_2	1.00	0.5254	0.5254
(4, 6)	s_1, s_2	s_1, s_2	1.00	0.5736	0.5736
(5, 6)	s_2, s_5	s_2	0.25	0.3562	0.0890
(6, 6)	s_2, s_3, s_5	s_2	0.25	0.1240	0.0077
...	...	...	...	...	...

refine the grid point scores to narrow down grid points that are most probably close to the current target location, we have redefined $w_{xy}(t)$ here to improve the accuracy for target location. The weight for the grid point $P(x, y)$ at time instant t is defined as,

$$w_{xy}(t) = \begin{cases} 0 & \text{if } S_{rep,xy}(t) = \{\phi\} \\ 4^{-\Delta k_{rep,xy}(t)} & \text{otherwise} \end{cases} \tag{11}$$

where $\Delta k_{rep,xy}(t)$ measures the degree of difference in the set of sensors that reported and those sensors that can detect point $P(x, y)$ at time instant t. The parameter $\Delta k_{rep,xy}(t)$ is defined as

$$\Delta k_{rep,\,xy}(t) = |k_{rep}(t) - k_{rep,xy}(t)| + |k_{rep}(t) - k_{xy}| \tag{12}$$

where $k_{xy} = |S_{xy}|$, $k_{rep}(t) = |S_{rep}(t)|$, and $k_{rep,\,xy}(t) = |S_{rep,xy}(t)|$. The parameter w_{xy} is therefore a decaying factor that is 1 only if $S_{rep}(t) = S_{xy}$. The number 4 in the formula for $w_{xy}(t)$ was chosen empirically after it was found to provide accurate simulation results. We are using $w_{xy}(t)$ to filter out grid points that are not likely to be close to the actual target location. The score is based on both the probability value from the probability table and the current relationship between $S_{rep}(t)$, $S_{rep,xt}(t)$ and S_{xy}. Table II gives some score calculation examples for the grid points in Figure 11 at the time instant t_{start}.

4.3 Selection of Sensors to Query

Assume that the maximum number of sensors that are allowed to report an event is k_{max}, and the set of the sensors selected by the cluster head for querying at time t is $S_q(t)$, $S_q(t) \subseteq S_{rep}(t) \subseteq \{s_1, s_2, \ldots, s_k\}$. To select the sensor to query based on the event reports and the localization procedure, we first note that for time instant t, if $k_{max} \geq k_{rep}(t)$, then all reported sensors can be queried. Otherwise, we select sensors based on a score-based ranking. The sensors selected correspond to the ones that have the shortest distance to those grid points with the highest scores. This selection rule is defined as

$$S_q(t) : d(S_q(t), P_{MS}) = \min\{d(s_i, P_{MS})\} \tag{13}$$

where $s_i \in S_{rep}(t)$, and P_{MS} denotes the set of grid points with the highest scores. Note it is possible that there are multiple grid points that have the maximum score. When this happens, we calculate the score concentration by averaging

Table III. Selected Sensors for the Example in Figure 10

t	$S_{rep}(t)$	$S_q(t)$	$\bar{S}_q(t)$	t	$S_{rep}(t)$	$S_q(t)$	$\bar{S}_q(t)$
...	...	...	...	...	...	...	...
3	s_1, s_2	s_1	s_2	4	s_2	s_2	s_2
5	s_2, s_3	s_3	s_2	6	s_2, s_3	s_3	s_2
7	s_2, s_3	s_3	s_3	8	s_3	s_3	s_3
...	...	...	...	...	...	...	...
16	s_4, s_5	s_4	s_4	17	s_4, s_5	s_4	s_5
18	s_2, s_3, s_5	s_2	s_2	19	s_2, s_5	s_5	s_2
20	s_1, s_2, s_5	s_2	s_2	21	s_5	s_5	s_5
...	...	...	...	...	...	...	...

the scores of the current grid point and its eight neighboring grid points. The grid point with the highest score (or the score concentration) is the most likely current target location. Therefore, selecting sensors that are closest to this point guarantee that the selected sensors can provide the most detailed and accurate data in response to the subsequent queries. Note target identification is not possible as at this stage since the cluster head has no additional information other than $S_{rep}(t)$. However, the selected sensors provide enough information in the subsequent stage to facilitate target identification. We evaluate the accuracy of this target localization procedure by calculating the distance between the grid point with the highest score and the actual target location. For the example of Figure 10, Table III gives some results for the selected sensor when the target is moving from "Start" ($t = 1$) to "End." We assume $k_{max} = 1$, and the target is moving at a constant speed. $\bar{S}_q(t)$ is the set of sensors that are closest to the actual location of the target at time t. The results show that $S_q(t)$ matches $\bar{S}_q(t)$ in many cases. The example does not illustrate the advantages of our proposed strategy since not many sensors are actually involved at the same time for target detection. However, we show later in Section 5 that the proposed algorithm performs very well when many sensors are involved in the target detection and reporting process.

4.4 Evaluation of Energy Savings

We next evaluate the energy saved by the proposed probabilistic localization approach. Suppose the sensor node has three basic energy consumption types—sensing, transmitting, and receiving, and these power values (energy per unit time) are E_s, E_t, and E_r, respectively. If we select all sensors that reported the target for querying, the total energy consumed for the event happening at time instant t can be evaluated using the following set of equations:

$$E_1(t) = k_{rep}(t)(E_t + E_r)T_1 \tag{14}$$

$$E_2(t) = (k_{rep}(t)E_r + E_t)T_2 \tag{15}$$

$$E_3(t) = k_{rep}(t)(E_t + E_r)T_3 \tag{16}$$

$$E_4(t) = E_s T_s \tag{17}$$

$$E(t) = E_1(t) + E_2(t) + E_3(t) + E_4(t) \tag{18}$$

$$E = \sum_{t=t_{start}}^{t_{end}} E(t) \tag{19}$$

where E_1 is the energy required for reporting the detection of an object, E_2 is the energy required for transmitting query information from the cluster head by broadcasting and for receiving this information at the sensor nodes, and E_3 is the energy required by sensor nodes being queried to send detailed information to the cluster head. The parameters T_1, T_2, and T_3 denote the lengths of time involved in the transmission and reception, which are directly proportional to the sizes of data for yes/no messages, control messages to query sensors, and the detailed sensor data transmitted to the cluster head. The parameter T_s is the time of sensing activity of sensors. The parameters E denotes the total energy in this case for target localization from t_{start} to t_{end}. For the proposed probabilistic localization approach, we calculate the total energy consumption $E^\star$ as follows:

$$E_1^*(t) = k_{rep}(t)(E_t + E_r)T_1 \tag{20}$$

$$E_2^*(t) = (k_q(t)E_r + E_t)T_2 \tag{21}$$

$$E_3^*(t) = k_q(t)(E_t + E_r)T_3 \tag{22}$$

$$E_4^*(t) = E_s T_s \tag{23}$$

$$E^*(t) = E_1^*(t) + E_2^*(t) + E_3^*(t) + E_4^*(t) \tag{24}$$

$$E^* = \sum_{t=t_{start}}^{t_{end}} E^*(t) \tag{25}$$

where $E_1(t)^* = E_1(t)$, $E_4^*(t) = E_4(t)$, and the total energy consumed is denoted by E^*. Therefore, the energy savings via the use of the probabilistic target localization algorithm is

$$\Delta E = E - E^* = C \sum_{t=t_{start}}^{t_{end}} (k_{rep}(t) - k_q(t)) \tag{26}$$

where $C = E_r T_2 + (E_t + E_r)T_3$ is a constant. Since $k_q(t)$ is always less than or equal to $k_{rep}(t)$, we have $\Delta E \geq 0$. Also, ΔE is monotonically nondecreasing with time. Figure 12 shows the energy saved for the target trace in Figure 10.

4.5 Procedural Description for Target Localization

Figure 13 shows the pseudocode of the procedure to generate the probability table for each grid point. Figure 14 shows the pseudocode for the simulation of the probabilistic localization algorithm. For an n by m grid with k sensors, the computational complexity involved in generating the probability table is $O(nm2^k)$ since the maximum number of sensors that can detect a grid point is k for the worst case. The computational complexity of the localization procedure is $O(nmk_{max})$, $k_{max} \leq k$. Therefore, the computational complexity of the probabilistic localization algorithm is $\max\{O(nmk_{max}), O(nm2^k)\} = O(nm2^k)$. Even though the worst-case complexity of the localization procedure is exponential

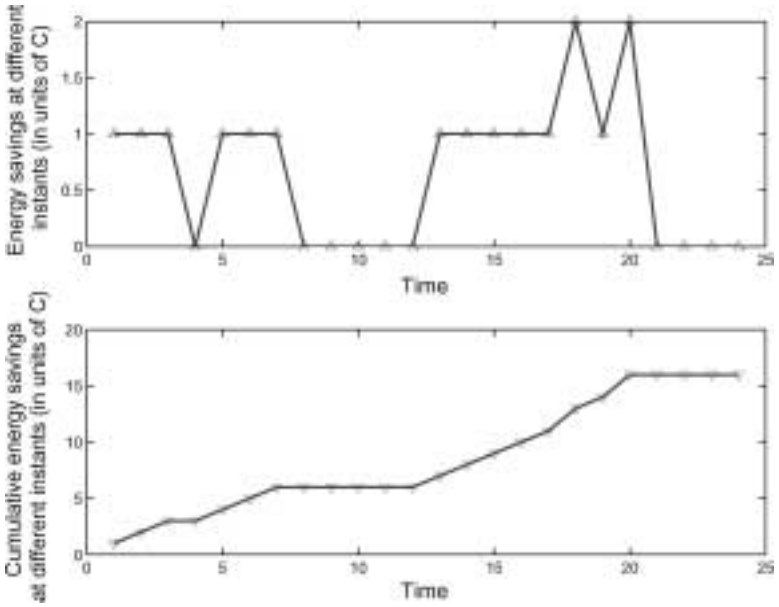

Fig. 12. Energy saved for the example in Figure 10.

Procedure *Generate_Probability_Table* (P(x, y), {s_1, s_2, . . . , s_k})

/* find S_{xy}, the set of sensors that can detect $P(x, y)$ */
For $s_i \in \{s_1, s_2, \ldots, s_k\}$
  If $d(s_i, P(x, y)) \leq r + r_e$
    $S_{xy} = S_{xy} \cup \{s_i\}$;
  End
End
/* fill up the probability table */
For i, $0 \leq i \leq k_{xy}$, $k_{xy} = |S_{xy}|$;
  If s_j detects $P(x, y)$
    Set $p_{xy}(s_j, i) = c_{x,y}(s_j)$;
  Else
    Set $p_{xy}(s_j, i) = 1 - c_{x,y}(s_j)$;
  End
  Set $p\_table_{xy}(i) = \prod_{s_j \in S_{xy}} p_{xy}(s_j, i)$;
End

Fig. 13. Pseudocodes for generating the probability table.

in k, in practice, the localization procedure can execute in less time since the number of sensors that can effectively detect a target at a given grid point is quite small.

5. SIMULATION RESULTS

In this section, we first present simulation results obtained using the VFA algorithm. The simulation results for the probabilistic localization algorithm are then presented using the sensor locations from the VFA algorithm as inputs.

Procedure *Localization* (Grid, $\{s_1, s_2, \ldots, s_k\}$, TargetTrace)

/* k_{max} is the maximum number of sensors that are allowed for querying, p_{rep} is the threshold level for a sensor to report to the cluster head of an event. *TargetTrace* starts from t_{start} and ends at t_{end}, with time unit as 1. */
1 Set $t = t_{start}$;
2 **While** $(t \leq t_{end})$
3 /* current target location */
4 Set $Target = TargetTrace(t)$;
5 /* calculate the scores */
6 Calculate $S_{rep}(t)$ from $\{s_1, s_2, \ldots, s_k\}$, $Target(t)$, p_{rep};
7 Set $k_{rep}(t) = |S_{rep}(t)|$;
8 **For** $P(x, y)$ in Grid, $x \in [1, width]$, $y \in [1, height]$
9 Set $k_{xy} = |S_{xy}|$;
10 Calculate $S_{rep,\,xy}(t)$ from $S_{rep}(t)$ and $P(x, y)$;
11 Calculate the index $i(t)$ of p_table_{xy} from $S_{rep}(t)$ and $S_{rep,\,xy}(t)$;
12 Set $k_{rep,\,xy}(t) = |S_{rep,\,xy}(t)|$;
13 **If** $S_{rep,xy}(t)=\{\phi\}$
14 $w_{xy}(t) = 0$;
15 **Else**
16 Set $\Delta k_{rep,xy}(t) = |k_{rep}(t) - k_{rep,xy}(t)| + |k_{rep}(t) - k_{xy}|$;
17 $w_{xy}(t) = 4^{-\Delta k_{rep,xy}(t)}$;
18 **End**
19 Set $SCORE_{xy}(t) = p\_table_{xy}(i(t)) \times w_{xy}(t)$;
20 **End**
21 /* select sensors for querying */
22 Calculate $S_q(t)$ from $SCORE_{xy}(t)$ and k_{max}, $x \in [1, width]$, $y \in [1, height]$;
23 /* next time instant */
24 Set $t = t + 1$;
25 **End**

Fig. 14. Pseudocode of the localization 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 1.0 GHz PC using Matlab.

5.1 Case Study 1: Binary Sensor Detection Model

Figures 15–18 present simulation results based on the binary sensor detection model. The initial locations of the sensors are shown in Figure 15. Figure 16 shows the final sensor positions determined by the VFA algorithm. For the binary sensor detection model, an 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 17 shows the virtual movement traces of all sensors during the execution of the VFA algorithm.

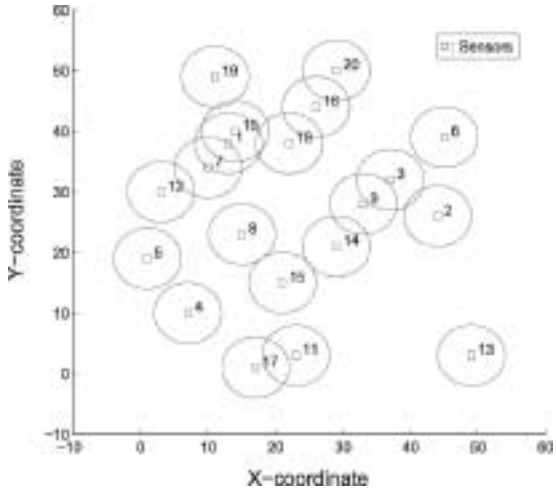

Fig. 15. Initial sensor positions after random placement (binary sensor detection model).

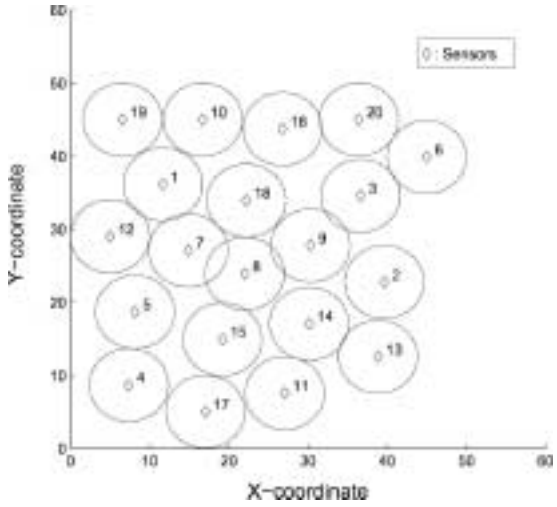

Fig. 16. Sensor positions after the execution of the VFA algorithm (binary sensor detection model).

Figure 18 shows the improvement in coverage during the execution of the VFA algorithm.

5.2 Case Study 2: Probabilistic Sensor Detection Model

Figures 19–21 present simulation results for the probabilistic sensor model. 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 19. Figure 20 shows the final sensor positions determined by the VFA algorithm. Figure 21 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 22

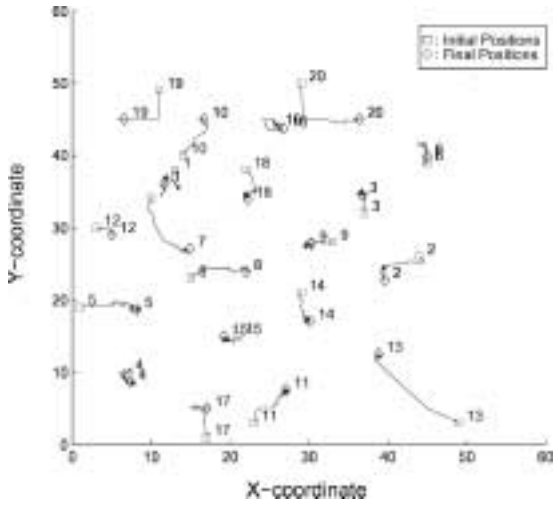

Fig. 17. A trace of virtual moves made by the sensors (binary sensor detection model).

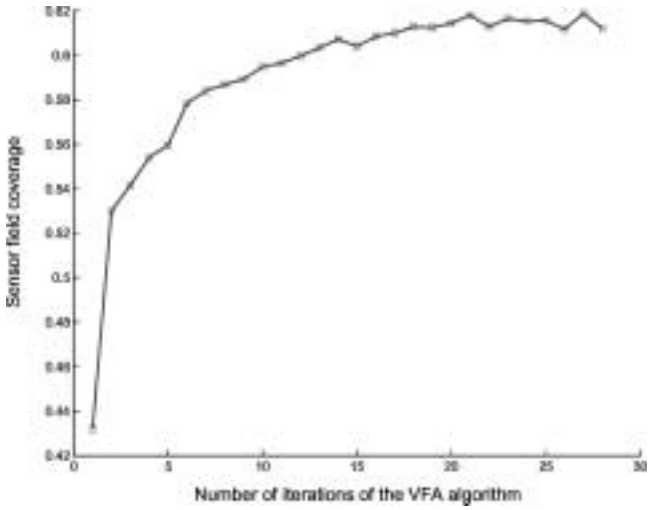

Fig. 18. Sensor field coverage improvement by the VFA algorithm (binary sensor detection model).

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 22 (roughly 0.38) is lower than the upper bound for the case of binary sensor detection model in Figure 18 (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.

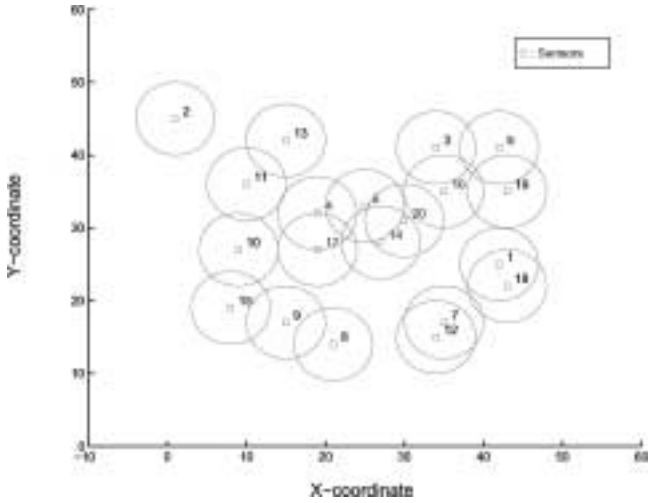

Fig. 19. Initial sensor positions after random placement (probabilistic sensor detection model).

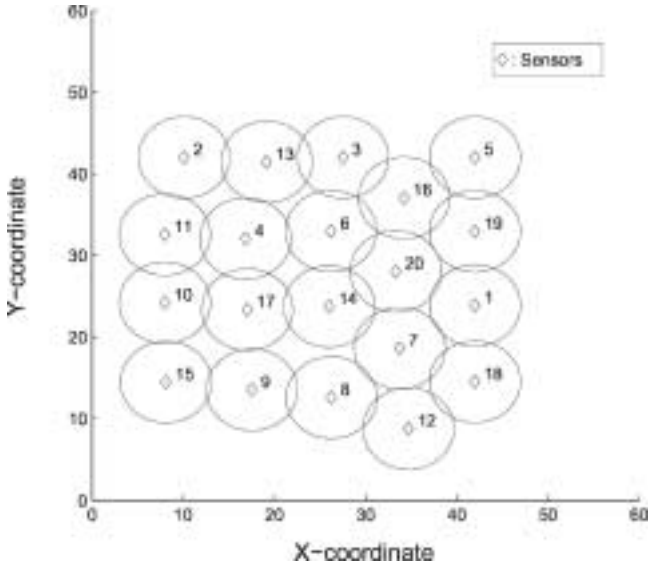

Fig. 20. Sensor positions after the execution of the VFA algorithm (probabilistic sensor detection model).

5.3 Case Study 3: Sensor Field with a Preferential Area and an Obstacle

As discussed in Section 3, the 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 23–26 present simulation results for a 50 by 50 sensor field that contains an obstacle and a preferential area. The initial sensor placements are shown in Figure 23. Figure 24 shows the final sensor positions

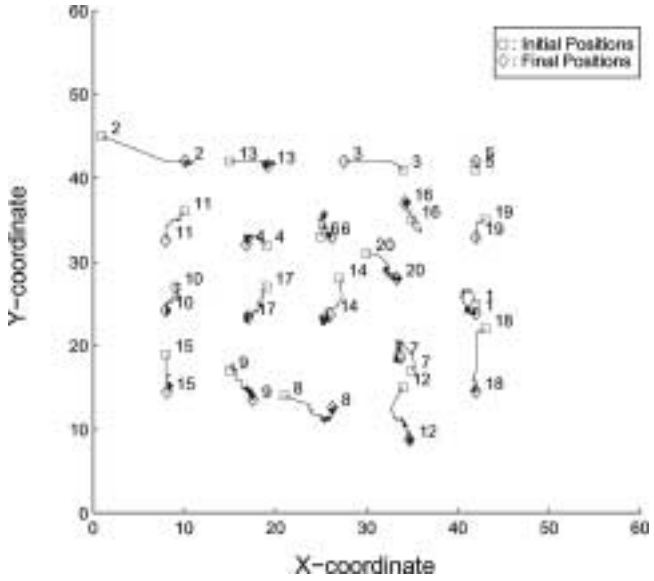

Fig. 21. A trace of virtual moves made by the sensors (probabilistic sensor detection model).

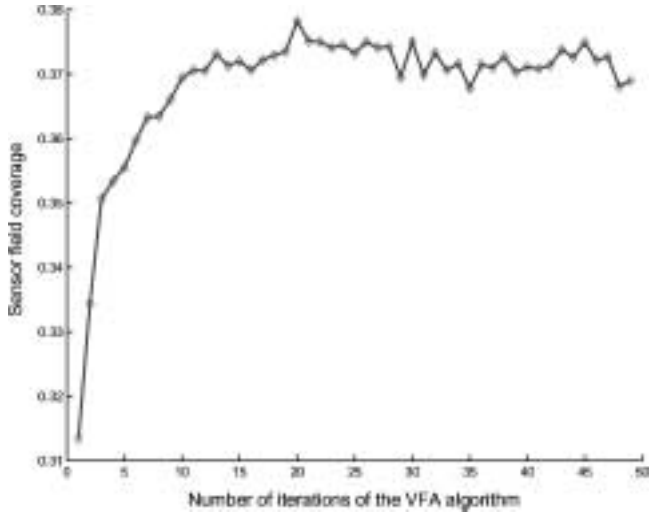

Fig. 22. Sensor field coverage achieved using the VFA algorithm (probabilistic sensor detection model).

determined by the VFA algorithm. Figure 25 shows the virtual movement traces of all sensors during the execution of the VFA algorithm. Figure 26 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 s for 30 iterations. For case study 2, the VFA algorithm took only 3 min to complete 50 iterations. Finally for case

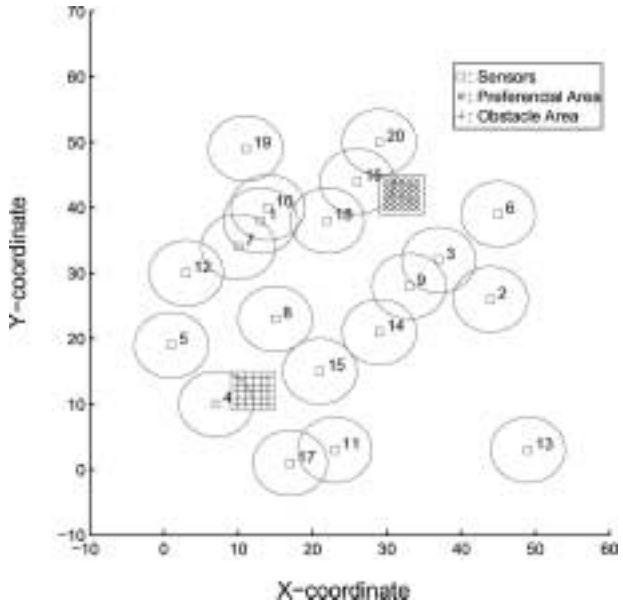

Fig. 23. Initial sensor positions after random placement with obstacles and preferred areas.

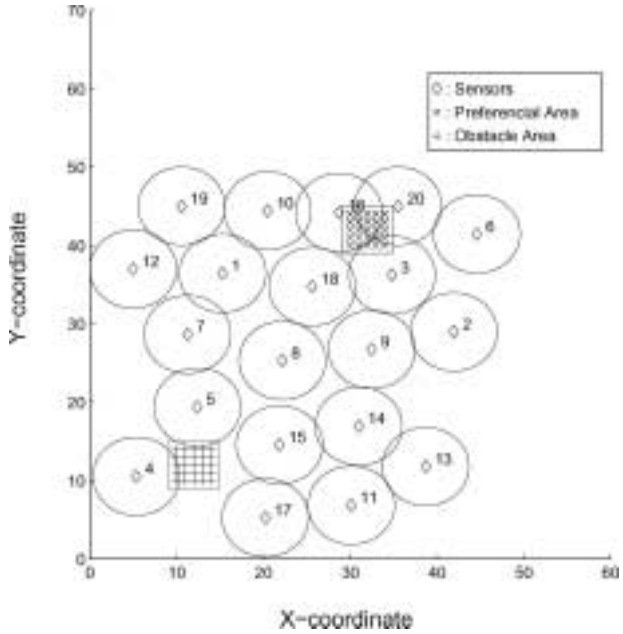

Fig. 24. Sensor positions after the execution of the VFA algorithm with obstacles and preferred areas.

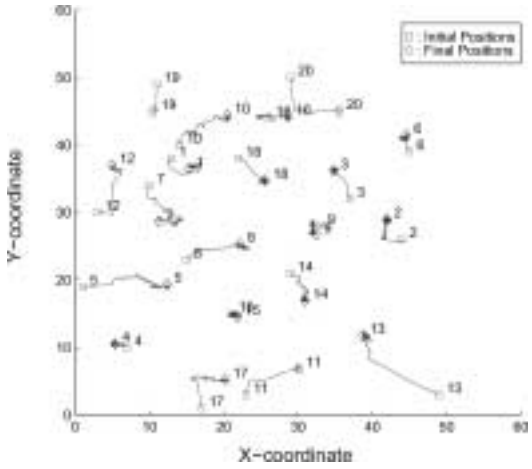

Fig. 25. A trace of virtual moves made by the sensors with obstacles and preferred areas.

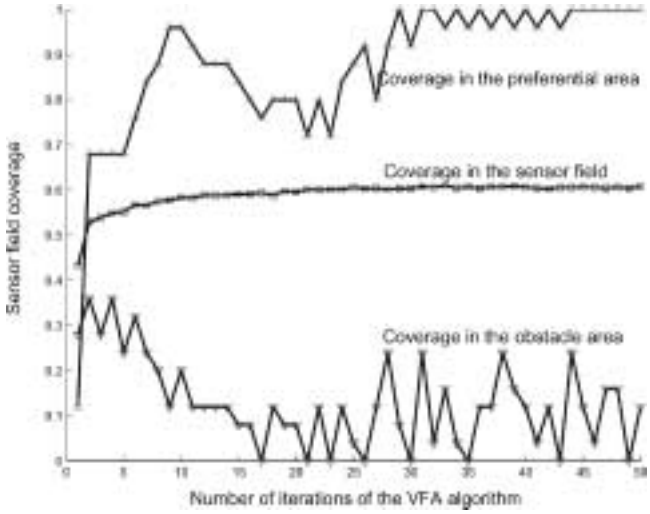

Fig. 26. Sensor field coverage achieved using the VFA algorithm with obstacles and preferred areas.

study in Section 3, the VFA algorithm took only 48 s 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 ten sets of random deployment results and take the average of the computation time. The results, listed in Table IV, 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 min on a Pentium III PC.

Table IV. The Computation Time for the VFA Algorithm for Larger Problem Instances

k	Binary Model (s)	Probability Model (min)	k	Binary Model (s)	Probability Model (min)
40	21	1.8	70	46	3.6
50	32	2.2	80	59	3.7
60	38	3.1	90	64	4.0

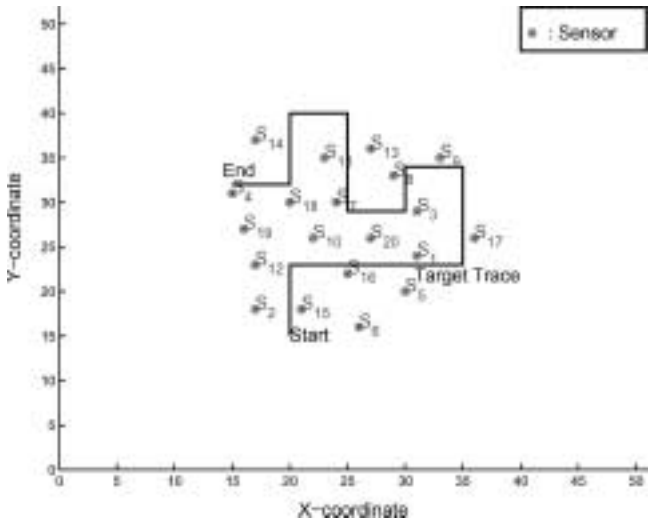

Fig. 27. Sensor field with sensors deployed by the VFA algorithm and target movement trace.

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

5.4 Case Study 4: Probability-Based Target Localization

We evaluate the localization algorithm using the results produced by the VFA algorithm in the sensor deployment stage. At this point, sensors are already moved to proper locations by the VFA algorithm. Figure 27 shows the sensor locations. There are total of 20 sensors deployed on a 50 by 50 sensor field grid, $r = 5$ grid units, $r_e = 3$ grid units, $c_{th} = 0.7$, $\lambda = 0.5$, and $\beta = 0.5$. To simulate target movement, we consider a target movement trace in the sensor grid as shown in Figure 27. The parameter t_{start} is the time instant that the target starts to move from its initial location marked as "Start" in Figure 27. Table V shows the results of the localization algorithm. We assume that a maximum of two sensors can be selected for querying by the cluster head. The target is assumed to move only 1 grid unit in one unit of time. There are total of 82 such moves in the simulated target movement trace. In the interest of conciseness, we only present the results for moves numbered 1–5, 41–45, and 78–82. The set $S_{rep}(t)$ indicates sensors that have reported the detection at time instant t. The set $S_q(t)$ includes sensors that are selected for querying by the cluster head at time t. The parameter $\Delta E(t)$ shows the energy saved by

Table V. Sensors Selected for Querying by the Cluster Head

t	$S_{rep}(t)$	$S_q(t)$	$\bar{S}_q(t)$	P_{MS}	$Target$	$\Delta E(t)$
01	s_2, s_6, s_{15}	s_2, s_{15}	s_2, s_{15}	(21, 15)	(20, 15)	C
02	$s_2, s_6, s_{12}, s_{15}, s_{16}$	s_2, s_{12}	s_2, s_{15}	(21, 15)	(20, 15)	$3C$
03	$s_2, s_6, s_{12}, s_{15}, s_{16}$	s_2, s_{12}	s_2, s_{15}	(21, 15)	(20, 15)	$3C$
04	$s_2, s_6, s_{12}, s_{15}, s_{16}$	s_2, s_{12}	s_2, s_{15}	(21, 15)	(20, 15)	$3C$
05	$s_2, s_6, s_{12}, s_{15}, s_{16}$	s_2, s_{12}	s_2, s_{15}	(21, 15)	(20, 15)	$3C$
...	...	...	...	...	...	...
41	s_3, s_8, s_9, s_{13}	s_8, s_9	s_8, s_9	(31, 34)	(31, 34)	$2C$
42	$s_3, s_7, s_8, s_9, s_{11}, s_{13}$	s_8, s_{13}	s_8, s_9	(28, 34)	(30, 34)	$4C$
43	$s_3, s_7, s_8, s_9, s_{11}, s_{13}, s_{20}$	s_{13}, s_8	s_8, s_9	(27, 34)	(30, 33)	$5C$
44	$s_3, s_7, s_8, s_9, s_{11}, s_{13}, s_{20}$	s_{13}, s_8	s_8, s_3	(27, 34)	(30, 32)	$5C$
45	$s_3, s_7, s_8, s_9, s_{11}, s_{13}, s_{20}$	s_{13}, s_8	s_8, s_3	(27, 34)	(30, 32)	$5C$
...	...	...	...	...	...	...
78	$s_4, s_7, s_{10}, s_{11}, s_{14}, s_{18}, s_{19}$	s_{18}, s_7	s_{18}, s_4	(27, 34)	(30, 32)	$5C$
79	$s_4, s_7, s_{10}, s_{11}, s_{14}, s_{18}, s_{19}$	s_{18}, s_7	s_{18}, s_4	(27, 34)	(30, 32)	$5C$
80	$s_4, s_7, s_{10}, s_{11}, s_{14}, s_{18}, s_{19}$	s_{18}, s_7	s_4, s_{18}	(27, 34)	(30, 32)	$5C$
81	$s_4, s_{11}, s_{14}, s_{18}, s_{19}$	s_4, s_{18}	s_4, s_{18}	(27, 34)	(30, 32)	$3C$
82	$s_4, s_{14}, s_{18}, s_{19}$	s_4, s_{19}	s_4, s_{19}	(27, 34)	(30, 32)	$2C$

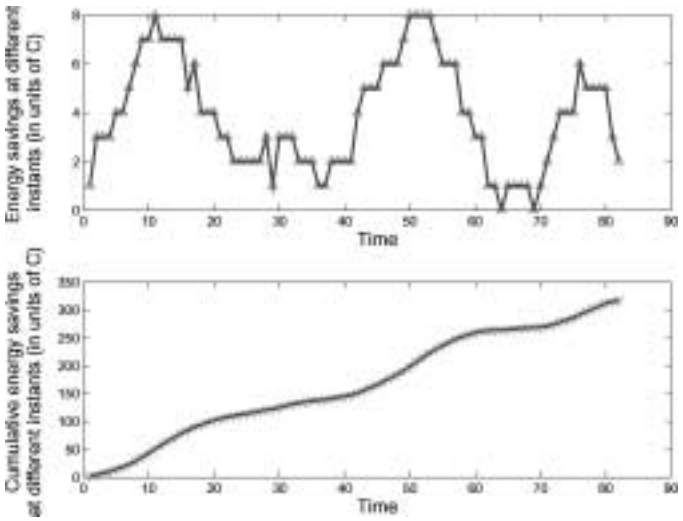

Fig. 28. Energy saving for target localization.

the localization algorithm for the detection event at time instant t. Figure 29 shows the estimated target location based on the grid point with the highest score. Figure 28 shows the energy saved during the target tracking process. Energy saved is evaluated in units of the constant C, given by Equation (26) in Section 4. The total computation time for generating the probability table is only 11 s. The total computation time for target localization for the total of 82 locations is only 16 s, with an average of 0.2 s per time instant.

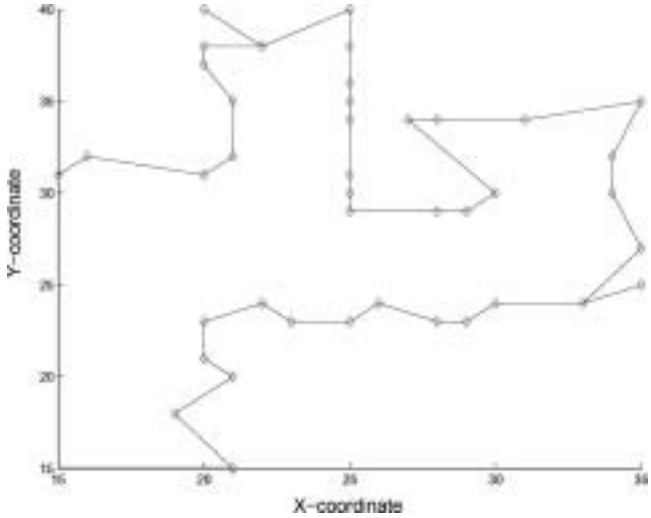

Fig. 29. Target localization by the grid point with the highest score.

5.5 Discussion

From the simulation results, we see that the VFA algorithm improves the sensor field coverage considerably compared to random sensor placement. The results of the proposed energy-conserving target localization method also show that considerable energy is saved in localizing a target. 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 base on the initial configuration. 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 for the DSN.

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. Finally, the proposed target localization algorithm can also be used for a deterministic sensor placement based on the precomputation of sensor locations.

6. CONCLUSION

In this paper, we have proposed the virtual force algorithm (VFA) as a practical approach for sensor deployment. The VFA algorithm uses a force-directed approach to improve the coverage provided by an initial random placement. The

VFA algorithm offers a number of important advantages. These include negligible computation time and a one-time repositioning of the sensors. Moreover, the desired sensor field coverage and model parameters can be provided as inputs to the VFA algorithm, thereby ensuring flexibility. We have shown how a probabilistic localization algorithm can be used in combination with force-directed sensor placement. We have also shown that the proposed probabilistic localization algorithm can significantly reduce the energy consumption for target detection and location.

Our future work will be focused on overcoming the current limitations of the VFA algorithm. The VFA algorithm can be made more efficient if it is provided with the theoretical bounds on the number of sensors needed to achieve a given coverage threshold. Also, there is no route plan for repositioning the sensors in the VFA algorithm, where sensor collision can happen during the repositioning. The VFA algorithm also requires accurate location information from the sensor nodes, it is better to consider a relaxed model with little requirements for the knowledge of all sensor nodes locations. Since the current target localization algorithm considers only one target in the sensor field, it is necessary to extend the proposed approach to facilitate scenarios for multiple objects localization. Extensions to nonmobile sensor nodes and situations of sensor node failures will also be considered in future work. Finally, we will examine continuous coordination systems instead of discrete coordination systems in this work.

ACKNOWLEDGMENTS

This research was supported in part by ONR under grant no. N66001-00-1-8946. It was also sponsored in part by DARPA, and administered by the Army Research Office under Emergent Surveillance Plexus MURI award no. DAAD19-01-1-0504. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies. The authors also thank the anonymous reviewers for their valuable comments.

REFERENCES

BHARDWAJ, M. AND CHANDRAKASAN, A. P. 2002. Bounding the lifetime of sensor networks via optimal role assignments. In *Proceedings of IEEE INFOCOM*, 1587–1596.

BROOKS, R. R. AND IYENGAR, S. S. 1997. *Multi-Sensor Fusion: Fundamentals and Applications with Software*. Prentice Hall, Upper Saddle River, NJ.

BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. 2001. Adaptive beacon placement. In *Proceedings of the International Conference on Distributed Computing Systems*, 489–498.

CHAKRABARTY, K., IYENGAR, S. S., QI, H., AND CHO, E. 2001. Coding theory framework for target location in distributed sensor networks. In *Proceedings of the International Symposium on Information Technology: Coding and Computing*, 130–134.

CHAKRABARTY, K., IYENGAR, S. S., QI, H., AND CHO, E. 2002. Grid coverage for surveillance and target location in distributed sensor networks. *IEEE Trans. Comput. 51*, 1448–1453.

DHILLON, S. S., CHAKRABARTY, K., AND IYENGAR, S. S. 2002. Sensor placement algorithms for grid coverage. In *Proceedings of the International Conference on Information Fusion*, 1581–1587.

ELFES, A. 1990. Occupancy grids: A stochastic spatial representation for active robot perception. In *Proceedings of the 6th Conference on Uncertainty in AI*, 60–70.

HEIDEMANN, J. AND BULUSU, N. 2001. Using geospatial information in sensor networks. In *Proceedings of CSTB Workshop on Intersection of Geospatial Information and Information Technology*.

HOWARD, A., MATARIĆ, M. J., AND SUKHATME, G. S. 2002. Mobile sensor network deployment using potential field: A distributed scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5: Proceedings of the 6th International Conference on Distributed Autonomous Robotic Systems (DARS02)*, 299–308.

IYENGAR, S. S., PRASAD, L., AND MIN, H. 1995. *Advances in Distributed Sensor Technology*. Prentice Hall, Englewood Cliffs, NJ.

KASETKASEM, T. AND VARSHNEY, P. K. 2001. Communication structure planning for multisensor detection systems. In *IEE Proceedings of Radar, Sonar and Navigation*. Vol. 148, 2–8.

LOCATELI, M. AND RABER, U. 2002. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Appl. Math. 122*, 139–166.

MEGUERDICHIAN, S., SLIJEPCEVIC, S., KARAYAN, V., AND POTKONJAK, M. 2001. Coverage problems in wireless ad-hoc sensor networks. In *Proceedings of IEEE INFOCOM*. Vol. 3, 1380–1387.

MUSMAN, S. A., LEHNER, P. E., AND ELSAESSER, C. 1997. Sensor planning for elusive targets. *J. Computer & Mathematical Modeling 25*, 103–115.

O'ROURKE, J. 1987. *Art Gallery Theorems and Algorithms*. Oxford University Press, New York.

PENNY, D. E. 1998. The automatic management of multi-sensor systems. In *Proceedings of the International Conference on Information Fusion*.

PRIYANTHA, N. B., CHAKRABORTY, A., AND BALAKRISHNAN, H. 2000. The cricket location-support system. In *Proceedings of ACM/IEEE International Conference on Mobile Computing and Networking*. 32–43.

QI, H., IYENGAR, S. S., AND CHAKRABARTY, K. 2001. Multi-resolution data integration using mobile agents in distributed sensor networks. *IEEE Trans. System, Man and Cybernetics 31*, 383–391.

RAPPAPORT, T. S. 1996. *Wireless Communications: Principles and Practice*. Prentice Hall, Upper Saddle River, NJ.

VARSHNEY, P. K. 1996. *Distributed Detection and Data Fusion*. Springer, New York.

ZOU, Y. AND CHAKRABARTY, K. 2003. Sensor deployment and target localization based on virtual forces. In *Proceedings of IEEE INFOCOM*.

Received September 2002; accepted January 2003