

# Sensor Deployment and Target Localization Based on Virtual Forces

Yi Zou and Krishnendu Chakrabarty

**Abstract**—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 virtual motion paths and the rate of movement for the randomly-placed sensors. Once the effective sensor positions are identified, a one-time movement with energy consideration incorporated is carried out, i.e., 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.

**Index Terms**—Sensor coverage, distributed sensor networks, sensor placement, virtual force, localization.

## I. 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, 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.

Y. Zou and K. Chakrabarty are with the Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA. E-mail: {yz1, krish}@ee.duke.edu.

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.

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 inspired by disk packing theory [11] and the virtual force field concept from robotics [5]. 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. 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, 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 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 III.

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

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

## II. RELATED PRIOR WORK

Sensor deployment problems have been studied in a variety of contexts [1], [2], [9]. 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 [12].

Sensor deployment and sensor planning for military applications are described in [6], where a general sensor model is used to detect elusive targets in the battlefield. However, the proposed DSN framework in [6] 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 sensing field has been considered for multi-robot exploration [5]. 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. 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 [7]. 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 [13]. 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 [13] 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 [3], [4]. 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, i.e. 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 [10].

A probabilistic optimization framework for minimizing the number of sensors for a two-dimensional grid has been proposed recently [10]. 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 [14]. Other related work includes the placement of a given number of sensors to reduce communication cost [15], optimal sensor placement for a given target distribution [16].

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.

## III. VIRTUAL FORCE ALGORITHM

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

### A. 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.
- 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 [5] and disk packing [11]. 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 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.

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  $d(s_i, P)$ , i.e.  $d(s_i, P) = \sqrt{(x_i - x)^2 + (y_i - y)^2}$ . Equation (1) shows the binary sensor model [3], [4] that expresses the coverage  $c_{xy}(s_i)$  of a grid point  $P$  by sensor  $s_i$ .

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

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 following, motivated in part by [8]:

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

where  $r_e (r_e < r)$  is a measure of the uncertainty in sensor detection,  $a = d(s_i, P) - (r - r_e)$ , and  $\alpha$  and  $\beta$  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 Fig. 1. Note that distances are measured in units of grid points. Fig. 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  $\alpha$  and  $\beta$  yield different translations reflected by different detection probabilities, which can be viewed as the characteristics of various types of physical sensors.

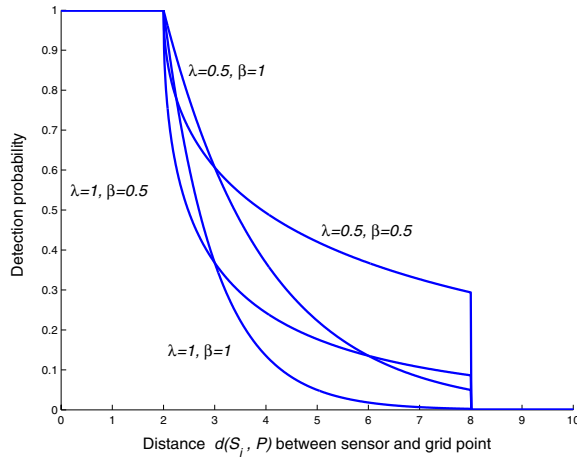


Fig. 1. Probabilistic sensor detection model.

### B. 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, 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.

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. 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

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

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  $\theta$  for vector  $\vec{f}$ .

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

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  $s_j$ ,  $\alpha_{ij}$  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 Fig. 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 Fig. 2.

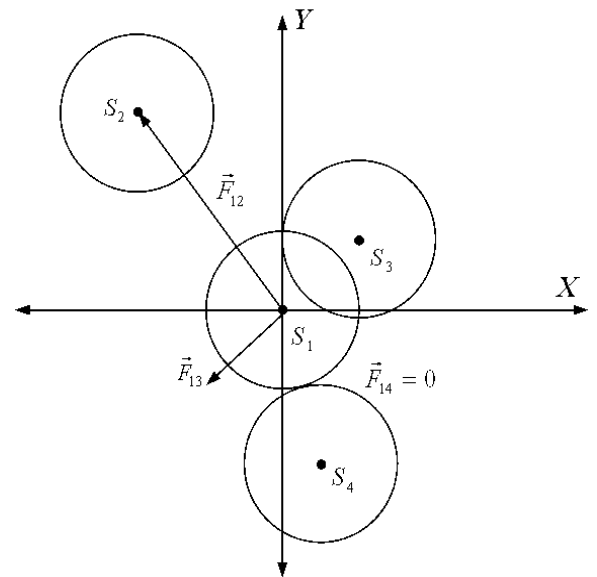


Fig. 2. An example of virtual forces with four sensors.

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 Fig. 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 Fig. 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.

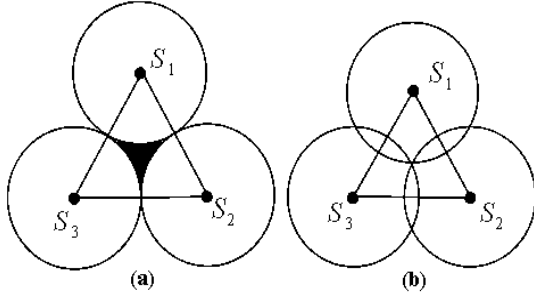


Fig. 3. Non-overlapped and overlapped sensor coverage areas.

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

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

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

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

Note that Equation (5) can also be extended to a region which is overlapped by a set of  $k_{ov}$  sensors, denoted as  $S_{ov}$ ,  $k_{ov} = |S_{ov}|$ ,  $S_{ov} \subseteq \{s_1, s_2, \dots, s_k\}$ . The coverage in this case is given by:

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

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 Fig. 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  $c_{th}$ .

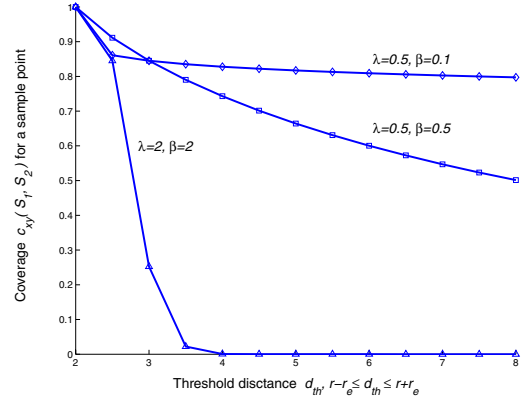


Fig. 4. Coverage vs.  $d_{th}$  of a sample point inside the overlapped area of  $s_1$  and  $s_2$ .

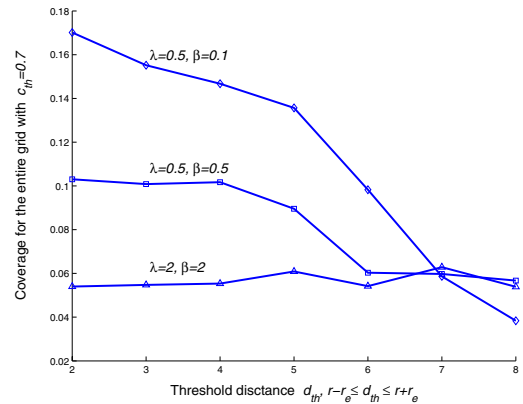


Fig. 5. Coverage vs.  $d_{th}$  with  $c_{th} = 0.7$  and different  $\lambda$  and  $\beta$ .

In order to prolong battery life, the distances between the initial and final position of the sensors are limited in the repositioning phase to conserve energy. We investigated two approaches for incorporating energy constraints in the VFA algorithm. The first approach disables any virtual forces on a sensor whenever the current distance reaches the distance limit. The second method records all virtual locations that sensors are moved into during the VFA algorithm. When the VFA algorithm terminates, a search procedure is used to find the locations with maximum coverage, except those locations that are already beyond the distance limit.

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

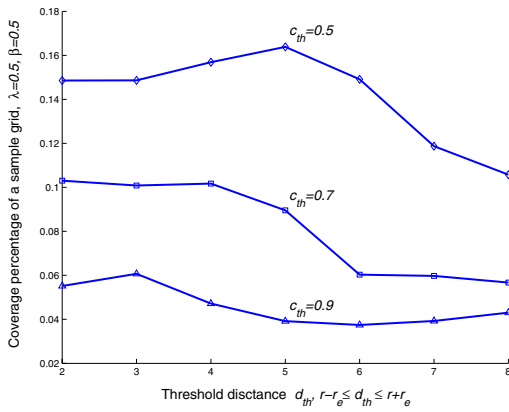


Fig. 6. Coverage vs.  $d_{th}$  with  $\lambda = 0.5$  and  $\beta = 0.5$  and  $c_{th} = 0.5, 0.7$  and  $0.9$ .

#### VFA Data Structures: $\text{Grid}, \{s_1, s_2, \dots, s_k\}$

```

/* 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. */
1 Grid structure:
2 Properties: width, height, k, c_th, d_th;
3 Preferential areas: PA_i(x, y, wx, wy),
  i = 1, 2, ..., n_P;
4 Obstacles areas: OA_i(x, y, wx, wy),
  i = 1, 2, ..., n_O;
5 Grid points, P_xy:
  c_xy(s_1, s_2, ..., s_k), S_xy, k_xy, p_table_xy;
6 Sensor s_i structure: i, (x, y), r, r_e, alpha, beta;

```

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

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.

We next describe the VFA algorithm in pseudo-code form. Fig. 7 shows the data structure of the VFA algorithm and Fig. 8 shows the implementation details. For a  $n$  by  $m$  grid with a total of  $k$  sensors deployed, the computational complexity of the VFA algorithm is  $O(nmk)$ .

#### IV. 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

#### Procedure *Virtual\_Force\_Algorithm* ( $\text{Grid}, \{s_1, s_2, \dots, s_k\}$ )

```

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, ..., 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     If coverage requirements are met
10      Break from While loop;
11    End
12  End
13  /* virtual forces among sensors */
14  For s_i in {s_1, s_2, ..., s_k}
15    Calculate F_ij using d(s_i, s_j), d_th, w_A, w_R;
16    Calculate F_iA using d(s_i, PA_1, ..., PA_n_P), d_th;
17    Calculate F_iR using d(s_i, OA_1, ..., OA_n_O), d_th;
18    F_i = sum F_ij + F_iR + F_iA, j in [1, k], j != i;
19  End
20  /* move sensors virtually */
21  For s_i in {s_1, s_2, ..., s_k}
22    F_i(s_i) virtually moves s_i to its next position;
23  End
24  Set loops = loops + 1;
25 End

```

Fig. 8. Pseudocode of the VFA algorithm.

queries the sensor(s) in the vicinity of the target. We assume here that the sensor detection reports are time-labeled.

#### A. 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  $S_{xy}$ ,  $|S_{xy}| = k_{xy}$ ,  $0 \leq k_{xy} \leq k$ , and  $S_{xy} \subseteq \{s_1, s_2, \dots, s_k\}$ . 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:

$$p\_table_{xy}(i) = \prod_{s_j \in S_{xy}} p_{xy}(s_j, i) \quad (8)$$

where  $0 \leq i \leq 2^{k_{xy}}$ , 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 3 sensors deployed.

Consider the grid point  $(2, 4)$  in Fig. 9 which is covered by all three sensors  $s_1, s_2$  and  $s_3$  with probabilities as 0.57, 1,

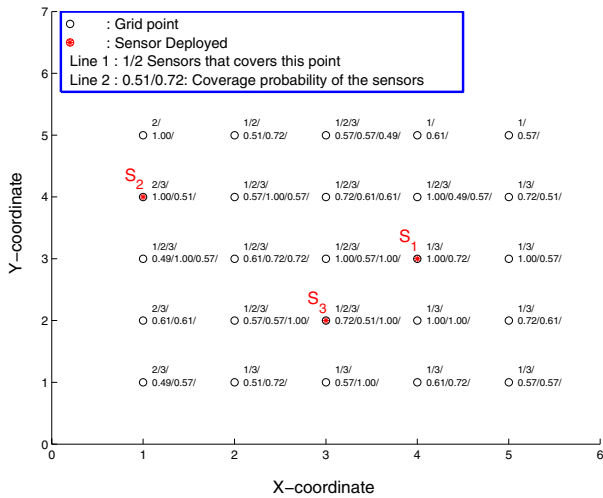


Fig. 9. Example of grid point probability table.

and 0.57 respectively. For the three sensors  $s_1$ ,  $s_2$  and  $s_3$ , there are a total of 8 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  $d_1 d_2 d_3$  ( $d_1, d_2, d_3 \in \{0, 1\}$ ) for a grid point, we calculate the conditional probabilities that the cluster head receives  $d_1 d_2 d_3$  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 number of entries in the detection probability tables for different grid points will in general be different.

TABLE I  
EXAMPLE PROBABILITY TABLE.

$i$	$d_1 d_2 d_3$	$p_{table_{xy}}(i)$	$i$	$d_1 d_2 d_3$	$p_{table_{xy}}(i)$
0	000	0	1	001	0
2	010	0.18	3	011	0.24
4	100	0	5	101	0
6	110	0.24	7	111	0.33

### B. 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 grid point 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 involves sending large amount of data, which consumes more energy consumption and needs more bandwidth. 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 Fig. 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”. Fig. 11 gives the score report at the time instant  $t_{start}$  when the target is present at “Start”.

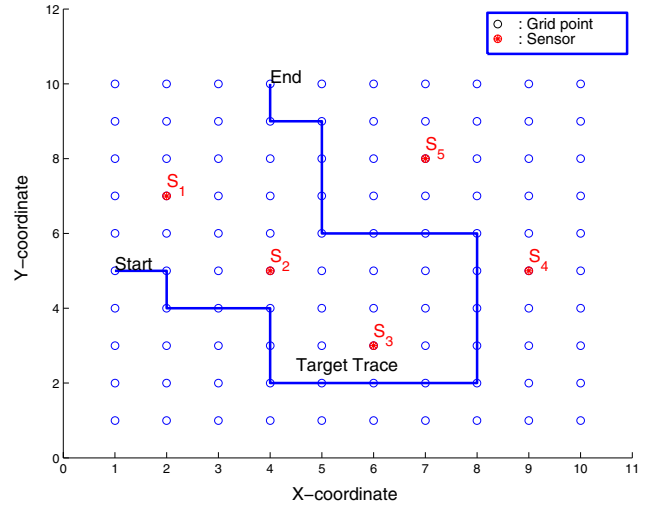


Fig. 10. Example sensor field with a moving target.

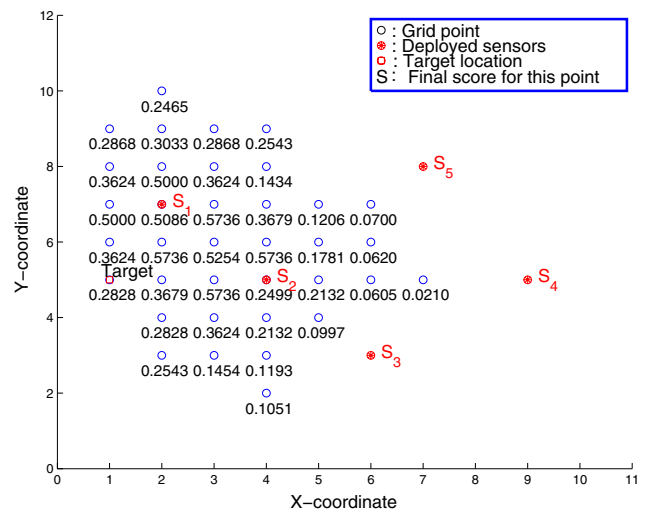


Fig. 11. Scoring results for target in the example sensor field at  $t_{start}$ .  $s_1$  and  $s_2$  have reported.

Assume  $S_{rep}(t)$  is the set of sensors that have reported the detection of an object,  $S_{rep,xy}(t)$  is the set of sensors that can detect point  $P(x, y)$  and have also reported the detection of an object. Obviously,  $S_{rep,xy}(t) \subseteq S_{rep}$  and  $S_{rep,xy}(t) \subseteq S_{xy}$ . We define the weight for the grid point  $P(x, y)$  at time instant  $t$  as,

$$w_{xy}(t) = \frac{k_{rep,xy}(t)}{k_{rep}(t)} \quad (9)$$

where  $k_{rep}(t) = |S_{rep}(t)|$  and  $k_{rep,xy}(t) = |S_{rep,xy}(t)|$ . The score of the grid point  $P(x, y)$  at time instant  $t$  is calculated as follows:

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

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  $P(x, y)$ . For example, consider grid point (1, 8) in Fig. 11, at time instant  $t_{start}$ ,  $p\_table_{18}(i(t_{start})) = 0.7248$ ,  $w_{18}(t_{start}) = 0.5$ , therefore  $SCORE_{18}(t_{start}) = 0.5 \times 0.7248 = 0.3624$ .

### C. 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  $S_q(t)$ ,  $S_q(t) \subseteq S_{rep}(t) \subseteq \{s_1, s_2, \dots, s_k\}$ . 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:

$$S_q(t) : d(S_q(t), P_{MS}) = \min\{d(s_i, P_{MS})\} \quad (11)$$

where  $s_i \in S_{rep}(t)$ , and  $P_{MS}$  denotes the set of grid points with the highest scores. For the example of Fig. 10, Table II shows the selected sensor when the target is moving from ‘‘Start’’ to ‘‘End’’. We assume here that a maximum of one sensor can be selected, and the target is moving at a constant speed. There are total of 24 locations for the target. We also assume the time instants are discrete, beginning with  $t = 1$ .

TABLE II  
SELECTED SENSORS FOR THE EXAMPLE IN FIG. 10.

$t$	$S_{rep}(t)$	$S_q(t)$	$t$	$S_{rep}(t)$	$S_q(t)$
1	$s_1, s_2$	$s_2$	17	$s_4, s_5$	$s_5$
2	$s_1, s_2$	$s_2$	18	$s_2, s_3, s_5$	$s_2$
3	$s_1, s_2$	$s_2$	19	$s_2, s_5$	$s_5$
4	$s_2$	$s_2$	20	$s_1, s_2, s_5$	$s_2$
5	$s_2, s_3$	$s_2$	21	$s_5$	$s_5$
...	...	...	...	...	...

### D. Evaluation of Energy Savings

We next evaluate the energy saved by the proposed probabilistic localization approach. Assume 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 equation:

$$E_1(t) = k_{rep}(t)(E_t + E_r)T_1 \quad (12)$$

$$E_2(t) = (k_{rep}(t)E_r + E_t)T_2 \quad (13)$$

$$E_3(t) = k_{rep}(t)(E_t + E_r)T_3 \quad (14)$$

$$E_4(t) = E_sT_s \quad (15)$$

$$E(t) = E_1(t) + E_2(t) + E_3(t) + E_4(t) \quad (16)$$

$$E = \sum_{t=t_{start}}^{t_{end}} E(t) \quad (17)$$

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 parameter  $E$  denotes the total energy in this case for target localization from  $t_{start}$  to  $t_{end}$ . Similarly, for the proposed probabilistic localization approach, we have:

$$E_1^*(t) = k_{rep}(t)(E_t + E_r)T_1 \quad (18)$$

$$E_2^*(t) = (k_q(t)E_r + E_t)T_2 \quad (19)$$

$$E_3^*(t) = k_q(t)(E_t + E_r)T_3 \quad (20)$$

$$E_4^*(t) = E_sT_s \quad (21)$$

$$E(t)^* = E_1(t)^* + E_2(t)^* + E_3(t)^* + E_4(t)^* \quad (22)$$

$$E^* = \sum_{t=t_{start}}^{t_{end}} E(t)^* \quad (23)$$

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:

$$\Delta E = E - E^* = C \sum_{t=t_{start}}^{t_{end}} (k_{rep}(t) - k_q(t)) \quad (24)$$

where  $C = E_rT_2 + (E_t + E_r)T_3$  is a constant. Since  $k_q(t)$  is always less than or equal to  $k_{rep}(t)$ , we have  $\Delta E \geq 0$ .

Fig. 12 shows the pseudocode of the procedure to generate the probability table for each grid point. Fig. 13 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  $max\{O(nmk_{max}), O(nm2^k)\} = O(nm2^k)$ .

---

**Procedure***Generate\_Probability\_Table* ( $P(x, y), \{s_1, s_2, \dots, s_k\}$ )

---

```
1 /* find  $S_{xy}$ , the set of sensors that can detect  $P(x, y)$  */
2 For  $s_i \in \{s_1, s_2, \dots, s_k\}$ 
3   If  $d(s_i, P(x, y)) \leq r + r_e$ 
4      $S_{xy} = S_{xy} \cup \{s_i\}$ ;
5   End
6 End
7 /* fill up the probability table */
8 For  $i, 0 \leq i \leq k_{xy}, k_{xy} = |S_{xy}|$ ;
9   If  $s_j$  detects  $P(x, y)$ 
10    Set  $p_{xy}(s_j, i) = c_{x,y}(s_j)$ ;
11  Else
12    Set  $p_{xy}(s_j, i) = 1 - c_{x,y}(s_j)$ ;
13  End
14  Set  $p\_table_{xy}(i) = \prod_{s_j \in S_{xy}} p_{xy}(s_j, i)$ ;
15 End
```

Fig. 12. Pseudocodes for generating the probability table.

---

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

---

```
/*  $k_{max}$  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.  $\text{TargetTrace}$  starts from  $t_{start}$  and it
ends at  $t_{end}$ . The simulation time unit is 1. */
1 Set  $t = t_{start}$ ;
2 While ( $t \leq t_{end}$ )
3   /* current target location */
4   Set  $\text{Target} = \text{TargetTrace}(t)$ ;
5   /* calculate the scores */
6   Calculate  $S_{rep}(t)$  from  $\{s_1, s_2, \dots, s_k\}, \text{Target}(t), p_{rep}$ ;
7   Set  $k_{rep}(t) = |S_{rep}(t)|$ ;
8   For  $P(x, y)$  in  $\text{Grid}, x \in [1, \text{width}], y \in [1, \text{height}]$ 
9     Calculate  $S_{rep,xy}(t)$  from  $S_{rep}(t)$  and  $P(x, y)$ ;
10    Calculate the index  $i(t)$  of  $p\_table_{xy}$ 
        from  $S_{rep}(t)$  and  $S_{rep,xy}(t)$ ;
11    Set  $k_{rep,xy}(t) = |S_{rep,xy}(t)|$ ;
12    Set  $w_{xy}(t) = \frac{k_{rep,xy}(t)}{k_{xy}(t)}$ ;
13    Set  $\text{SCORE}_{xy}(t) = p\_table_{xy}(i(t)) \times w_{xy}(t)$ ;
14  End
15  /* select sensors for querying */
16  Calculate  $S_q(t)$  from  $\text{SCORE}_{xy}(t)$  and  $k_{max}$ ,
         $x \in [1, \text{width}], y \in [1, \text{height}]$ ;
17  /* next time instant */
18  Set  $t = t + 1$ ;
19 End
```

Fig. 13. Pseudocode of the localization algorithm.

## V. SIMULATION RESULTS

In this section, we first present simulation results obtained using the VFA algorithm. Then the simulation results of the probabilistic localization algorithm are presented using the sensor location data from the VFA algorithm as inputs. 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 as 5 units ( $r = 5$ ), and range detection error as 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.

### A. Case Study 1: Binary Sensor Detection Model

Figures 14-16 present simulation results based on the binary sensor detection model. The initial locations of the sensors are shown in Fig. 14. Fig. 15 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. Fig. 16 shows the improvement in coverage during the execution of the VFA algorithm.

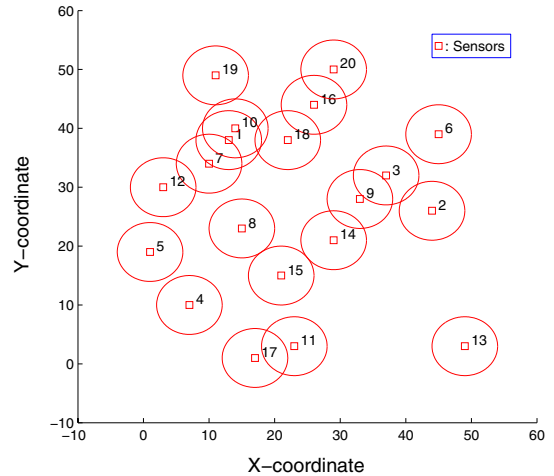


Fig. 14. Initial sensor positions after random placement (binary sensor detection model).

### B. Case Study 2: Probabilistic Sensor Detection Model

Figures 17-19 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 Fig. 17. Fig. 18 shows the final sensor positions determined by the VFA algorithm. Fig. 19 shows the virtual movement traces of all sensors during the execution of the VFA algorithm. We can see overlap areas are used to increase the number of grid points whose coverage exceeds the required threshold  $c_{th}$ .

### C. Case Study 3: Sensor Field with a Preferential Area and an Obstacle

As discussed in Section III, VFA is also applicable with sensor field containing obstacles and preferential areas. Obstacles should be avoided, therefore they are modeled as repulsive force sources in the VFA algorithm. Preferential areas should



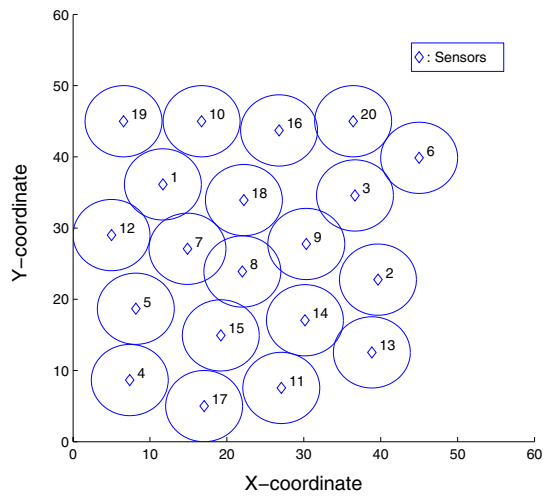


Fig. 15. Sensor positions after the execution of the VFA algorithm (binary sensor detection model).

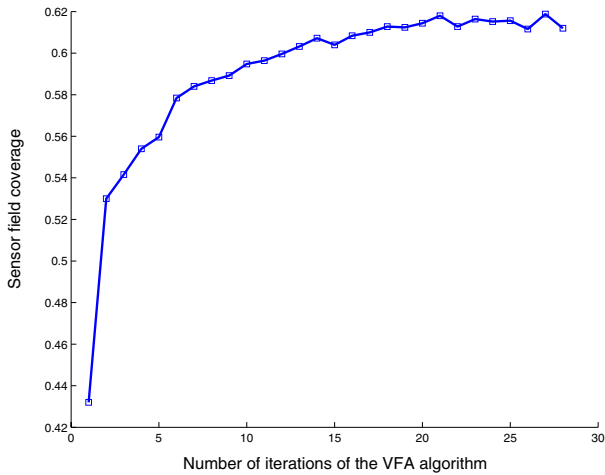


Fig. 16. Sensor field coverage improvement by the VFA algorithm (binary sensor detection model).

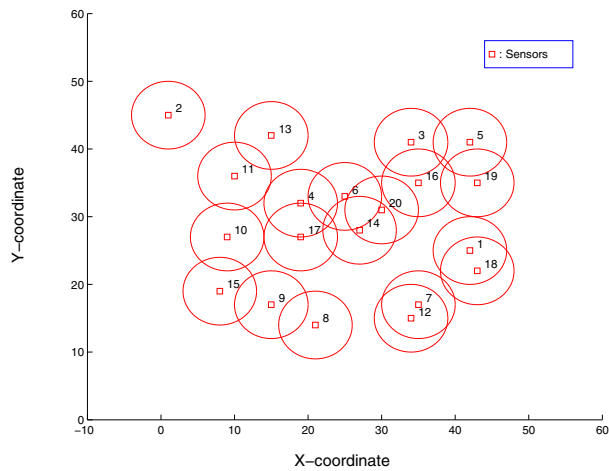


Fig. 17. Initial sensor positions after random placement (probabilistic sensor detection model).

be covered first, therefore they are modeled as attractive force sources in the VFA algorithm. Fig. 20-22 present simulation

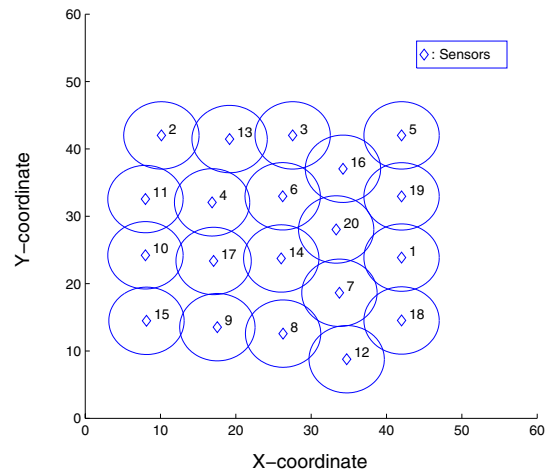


Fig. 18. Sensor positions after the execution of the VFA algorithm (probabilistic sensor detection model).

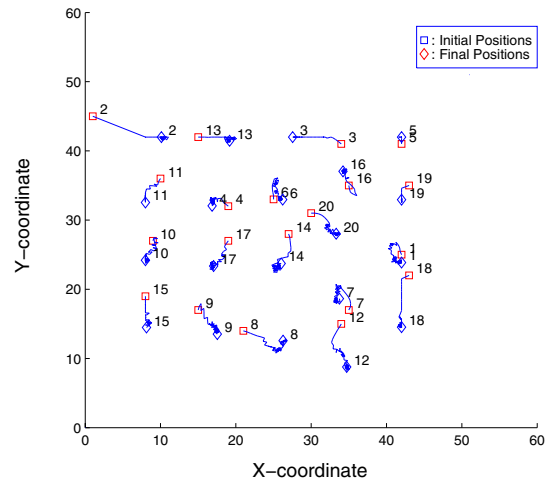


Fig. 19. A trace of virtual moves made by the sensors (probabilistic sensor detection model).

results for a 50 by 50 sensor field that contains an obstacle and a preferential area. The initial sensor placements are shown in Fig. 20. Fig. 21 shows the final sensor positions determined by the VFA algorithm. Fig. 22 shows the improvement of coverage during the execution of the VFA algorithm.

#### D. 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 stage, sensors are already moved to proper locations by the VFA algorithm. Fig. 23 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,  $c_{th} = 0.7$ ,  $\alpha = 0.5$ , and  $\beta = 0.5$ . To simulate the target movement, we consider a target movement trace in the sensor grid as shown in Fig. 23.  $t_{start}$  is the time instant that target starts to move from its initial location marked as “Start” in Fig. 23.

Table III shows the results of the localization algorithm. We assume that a maximum of two sensors can be selected for

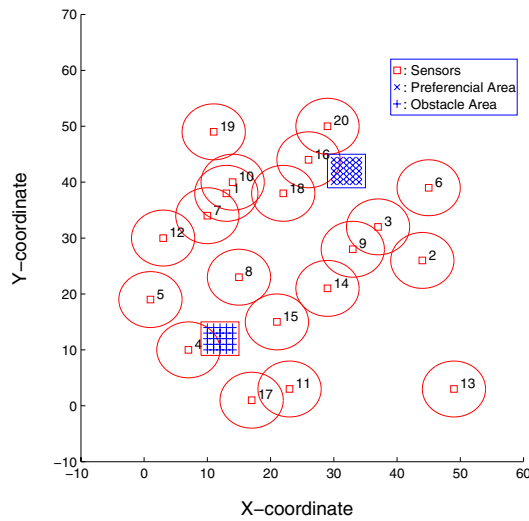


Fig. 20. Initial sensor positions after random placement with obstacles and preferred areas.

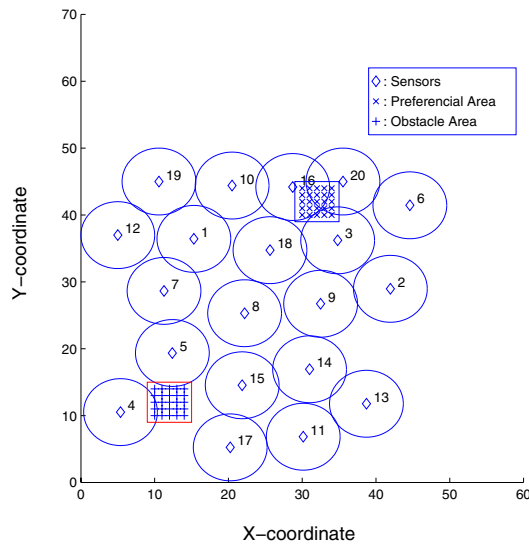


Fig. 21. Sensor positions after the execution of the VFA algorithm with obstacles and preferred areas.

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. Due to limited space, we only present the results for moves numbered 1-4, 41-44 and 79-82. 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 the localization algorithm for the detection event at time instant  $t$  in units of the constant  $C$ , defined in Section IV; see Equation (24).

### E. Discussion

From the simulation results, we see that the VFA algorithm improves the sensor field coverage considerably compared to random sensor placement, and it does not require much computation time. For Case Study 1, the VFA algorithm took

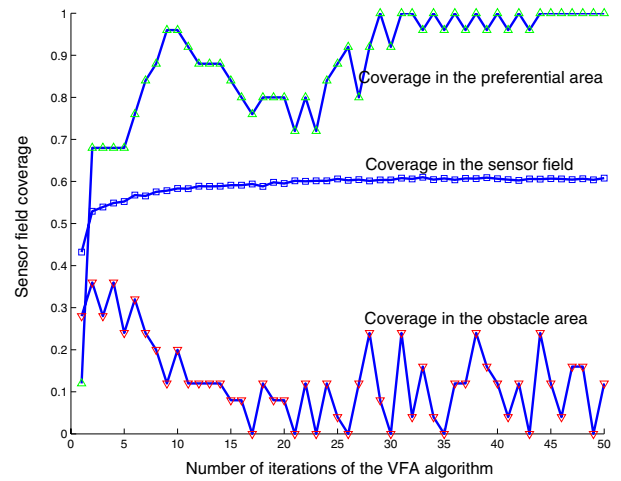


Fig. 22. Sensor field coverage achieved using the VFA algorithm with obstacles and preferred areas.

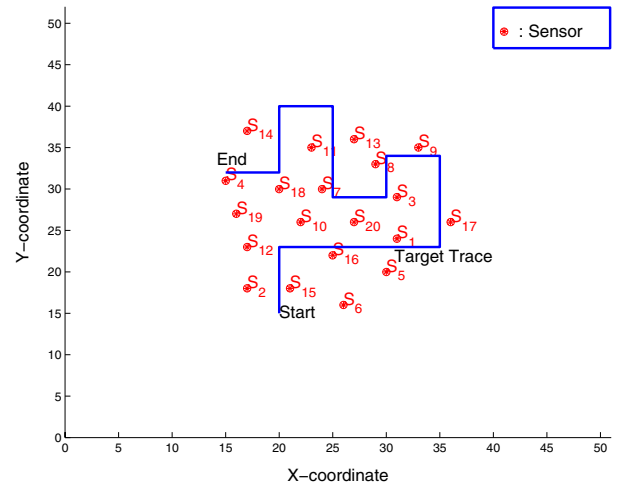


Fig. 23. Sensor field with sensors deployed by the VFA algorithm and target movement trace.

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 in Section 3, the VFA algorithm took only 48 seconds to complete 50 iterations. Note that these computation time include the time needed for displaying the simulation results on the screen. CPU time is important because sensors must be redeployed in real-time in the field. 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 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 proper rectangles. The VFA algorithm however is also applicable for alternative location indicators, distance measures, and models of preferential ar-

TABLE III

SENSORS SELECTED FOR QUERYING BY THE CLUSTER HEAD.

$t$	$S_{rep}(t)$	$S_q(t)$	$\Delta E(t)$
01	$s_2, s_6, s_{15}$	$s_2, s_{15}$	$C$
02	$s_2, s_6, s_{12}, s_{15}, s_{16}$	$s_2, s_{12}$	$3C$
03	$s_2, s_6, s_{12}, s_{15}, s_{16}$	$s_2, s_{12}$	$3C$
04	$s_2, s_6, s_{12}, s_{15}, s_{16}$	$s_2, s_{12}$	$3C$
...	...	...	...
41	$s_3, s_8, s_9, s_{13}$	$s_9, s_3$	$2C$
42	$s_3, s_7, s_8, s_9, s_{11}, s_{13}$	$s_9, s_3$	$4C$
43	$s_3, s_7, s_8, s_9, s_{11}, s_{13}, s_{20}$	$s_9, s_3$	$5C$
44	$s_3, s_7, s_8, s_9, s_{11}, s_{13}, s_{20}$	$s_9, s_3$	$5C$
...	...	...	...
79	$s_4, s_7, s_{10}, s_{11}, s_{14}, s_{18}, s_{19}$	$s_7, s_{11}$	$5C$
80	$s_4, s_7, s_{10}, s_{11}, s_{14}, s_{18}, s_{19}$	$s_7, s_{11}$	$5C$
81	$s_4, s_{11}, s_{14}, s_{18}, s_{19}$	$s_{11}, s_{18}$	$3C$
82	$s_4, s_{14}, s_{18}, s_{19}$	$s_{18}, s_{14}$	$2C$

eas 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.

## VI. 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. Since the current target localization algorithm considers only one target in the sensor field, it is necessary to extend the proposed approach to facilitate the localization of multiple objects. Another extension lies in distributed localization and querying. Extensions to non-mobile 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.

## ACKNOWLEDGEMENT

The authors thank Thomas Clouqueur of University of Wisconsin, and the anonymous reviewers for their valuable comments on the submitted version of this manuscript.

## REFERENCES

- [1] H. Qi, S. S. Iyengar, and K. Chakrabarty, "Multi-resolution data integration using mobile agents in distributed sensor networks", *IEEE Transactions on System, Man and Cybernetics (Part C)*, vol. 31, pp. 383-391, August 2001.
- [2] S. S. Iyengar, L. Prasad and H. Min, *Advances in Distributed Sensor Technology*, Prentice Hall, Englewood Cliffs, NJ, 1995.
- [3] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks", *IEEE Transactions on Computers*, vol. 51, pp. 1448-1453, 2002.
- [4] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho, "Coding Theory Framework for Target Location in Distributed Sensor Networks", *Proc. International Symposium on Information Technology: Coding and Computing*, pp.130-134, 2001.
- [5] A. Howard, M. J. Mataric and G. S. Sukhatme, "Mobile Sensor Network Deployment Using Potential Field: a distributed scalable solution to the area coverage problem", to appear in *Proc. International Conference on Distributed Autonomous Robotic Systems*, June 2002.
- [6] S. A. Musman, P. E. Lehner and C. Elsaesser, "Sensor Planning for Elusive Targets", *Journal of Computer & Mathematical Modeling*, vol. 25, No. 3, pp. 103-115, 1997.
- [7] S. Meguerdichian, S. Slijepcevic, V. Karayan and M. Potkonjak, "Coverage problems in wireless ad-hoc sensor networks", *Proc. IEEE Infocom*, vol. 3, pp. 1380-1387, 2001.
- [8] A. Elfes, "Occupancy Grids: A Stochastic Spatial Representation for Active Robot Perception", *Proc. 6th Conference on Uncertainty in AI*, pp. 60-70, July 1990.
- [9] R. R. Brooks and S. S. Iyengar, *Multi-Sensor Fusion: Fundamentals and Applications with Software*, Prentice Hall, Upper Saddle River, NJ, 1997.
- [10] S. S. Dhillon, K. Chakrabarty and S. S. Iyengar, "Sensor placement algorithms for grid coverage", *Proc. International Conference on Information Fusion*, pp. 1581-1587, 2002.
- [11] M. Locateli and U. Raber, "Packing equal circles in a square: a deterministic global optimization approach", *Discrete Applied Mathematics*, vol. 122, pp. 139-166, October 2002.
- [12] N. Bulusu, J. Heidemann and D. Estrin, "Adaptive beacon placement", *Proc. International Conference on Distributed Computing Systems*, pp. 489-498, 2001.
- [13] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, "The cricket location-support system", *Proc. ACM/IEEE International Conference on Mobile Computing and Networking*, pp. 32-43, 2000.
- [14] J. O'Rourke, *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, NY, 1987.
- [15] T. Kasetkasem and P. K. Varshney, "Communication structure planning for multisensor detection systems", *IEE Proc. Radar, Sonar and Navigation*, vol. 148, pp. 2-8, February 2001.
- [16] D. E. Penny, "The automatic management of multi-sensor systems", *Proc. International Conference on Information Fusion*, July 1998.