Application of Virtual Force Algorithm for Sensor Deployment

Ashish Singh Chauhan¹ Varun Kumar Pandey²

¹²Department of Electronics and Communication Engineering
Bhabha Institute of Technology Kanpur (D), India

Abstract—The impact of cluster-based distributed sensor networks relies to a great extent on the coverage provided by the deployment of these sensors. 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 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.

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.

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

II. VIRTUAL FORCE ALGORITHM

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

A. Preliminaries:

For 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 [1] and disk packing [5]. 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 \( s_i \).

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

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]:

\[
c_{xy}(s_i) = \begin{cases} 0, & \text{if } r + r_e \leq d(s_i, P) \\ e^{-\alpha d(s_i, P)}, & \text{if } r - r_e < d(s_i, P) < r + r_e \\ 1, & \text{if } r - r_e \geq d(s_i, P) \end{cases}
\]

where \( r_e \) 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 \( F_i \). Note that \( F_i \) 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 \( F_{ij} \).

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 buildings 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 \( F_{iO} \) be the total (attractive) force on \( s_i \) due to preferential coverage areas, and let \( F_{iR} \) be the total (repulsive) force on \( s_i \) due to obstacles. The total force \( F_i \) on \( s_i \) can now be expressed as
Application of Virtual Force Algorithm for Sensor Deployment
(IJSRD/Vol. 3/Issue 02/2015/414)

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 \( k_r^2 \) that can be covered with the \( k \) sensors exceeds the area of the grid.

Fig. 3: Non-Overlapped and Overlapped Sensor Coverage Areas

If \( r_e \approx 0 \) and we use the binary sensor detection model given by Equation (1), we attempt to make \( d_{ij} \) as close to 2\( r \) 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.

Fig. 4: An example of virtual forces with four sensors.

If \( r_e \approx 0 \) and we use the sensor detection model given by Equation (1), we attempt to make \( d_{ij} \) as close to 2\( r \) 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 \( k_r^2 \) that can be covered with the \( k \) sensors exceeds the area of the grid.

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

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 \( s_j \), \( \theta_{ij} \) is the orientation (angle) of a line segment from \( s_i \) to \( s_j \), and \( w_A(\theta) \) 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 \( \vec{F}_i \) on \( s_i \) is given by \( \vec{F}_i = \vec{F}_{i1} + \vec{F}_{i2} + \vec{F}_{i3} + \vec{F}_{i4} \). If we assume that \( d_{12} > d_{0h} \), \( d_{13} < d_{0h} \), and \( d_{14} = d_{0h} \), \( s_2 \) 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.

\[
F_i = \sum_{j=1, j \neq i}^k \vec{F}_{ij} + \vec{F}_{iR} + \vec{F}_{iA}
\]

(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 \( \vec{F} \).

\[
\vec{F}_{ij} = \begin{cases} 
(w_A(d_{ij} - d_{0h}), \alpha_{ij}) & \text{if } d_{ij} > d_{0h} \\
(0, w_A(\alpha_{ij} + \pi)) & \text{if } d_{ij} = d_{0h} \\
0 & \text{if otherwise}
\end{cases}
\]

(4)

Figures 4-6 show how the coverage is affected by \( d_{th} \) and
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 $c_{th}$.

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 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 $n \times m$ grid with a total of $k$ sensors deployed, the computational complexity of the VFA algorithm is $O(nmk)$.

### III. 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. We assume here that the sensor detection reports are time-labeled.
A. 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 $e = 1$. The zigzag shaped line is the target movement trace. The target starts to move at $t = t_{\text{start}}$ from the grid point marked as “Start” and finishes at $t = t_{\text{end}}$ at the grid point marked as “End”. Fig. 11 gives the score report at the time instant $t_{\text{start}}$ when the target is present at “Start”.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$p_{\text{Table}_{xy}}(i)$</th>
<th>$x$</th>
<th>$y$</th>
<th>$p_{\text{Table}_{xy}}(i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.01</td>
<td>1</td>
<td>0</td>
<td>0.01</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0.18</td>
<td>2</td>
<td>1</td>
<td>0.11</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0.10</td>
<td>3</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0.24</td>
<td>4</td>
<td>1</td>
<td>0.11</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.24</td>
<td>5</td>
<td>0</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Table 1: Example Probability Table
Application of Virtual Force Algorithm for Sensor Deployment  
(IJSRD/Vol. 3/Issue 02/2015/414)

**Fig. 10**: Example Sensor Field with a Moving Target

**Fig. 11**: Scoring Result for Target in the Example Sensor

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,

$$w_{xy}(t) = \frac{k_{rep,xy}(t)}{k_{rep}(t)}$$  \hspace{1cm} (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:

$$SCORE_{xy}(t) = p_{table}(s_i(t)) \times w_{xy}(t) \hspace{1cm} (10)$$

where $s_i(t)$ is the index of the $p_{table}$ at time $t$. The parameter $t$ is calculated from $S_{xy}$ and $S_{rep,xy}$. The parameter $p_{table}(s_i(t))$ 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 $t_{start} = p_{table}(s_i(t_{start})) = 0.7248$, $w_{xy}(t_{start}) = 0.5$, therefore $SCORE_{xy}(t_{start}) = 0.5 \times 0.7248 = 0.3624$.

**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 $S_q(t)$. $S_q(t) \subseteq S_{rep}(t) \subseteq \{s_1, s_2, \ldots, s_n\}$. 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:

$$S_q(t) : d(S_q(t), P_{MESS}) = \min \{d(s_i, P_{MESS})\} \hspace{1cm} (11)$$

where $s_i \in S_{rep}(t)$, and $P_{MESS}$ 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**

<table>
<thead>
<tr>
<th>t</th>
<th>$S_{rep}(t)$</th>
<th>$S_q(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$s_1, s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>2</td>
<td>$s_1, s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>3</td>
<td>$s_1, s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>4</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>5</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>6</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>7</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>8</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>9</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>10</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>11</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>12</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>13</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>14</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>15</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>16</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>17</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>18</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>19</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>20</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>21</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>22</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>23</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
<tr>
<td>24</td>
<td>$s_2, s_3$</td>
<td>$s_3$</td>
</tr>
</tbody>
</table>

**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...
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 $t_{end}$. Similarly, for the proposed probabilistic localization approach, we have:

$$
E_1(t) = k_{rep}(t)(E_1 + E_2)T_1
$$

$$
E_2(t) = (k_{rep}(t)E_2 + E_1)T_2
$$

$$
E_3(t) = k_{rep}(t)(E_1 + E_2)T_3
$$

$$
E(t) = E_1(t) + E_2(t) + E_3(t) + E_4(t)
$$

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

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 $O(nm^2k)$ 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 $O(nmmax)$, $kmax \leq k$. Therefore, the computational complexity of the probabilistic localization algorithm is max $O(nmmax)$, $O(nm^2k) = O(nm^2k)$.

**Procedure**

*Generate_Probability_Table* ($P(x, y), \{s_1, s_2, \ldots, s_k\}$)

1. /* find $S_{xy}$, the set of sensors that can detect $P(x, y)$ */
2. For $s_i \in \{s_1, s_2, \ldots, s_k\}$
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 $p_{xy}(s_j, i) = c_{xy}(s_j)$
11. Else
12. Set $p_{xy}(s_j, i) = 1 - c_{xy}(s_j)$
13. End
14. Set $p_{xy}(s_j, i) = \prod_{s \in S_{xy}} p_{xy}(s, i)$
15. End

Fig. 12: Pseudocodes for generating the probability table.

**Procedure**

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

/* $k_{total}$ 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 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 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. Calculate $S_{rep}xy(t)$ from $S_{rep}(t)$ and $P(x, y)$.
10. Calculate the index $i(t)$ of $p_{table}xy(t)$ from $S_{rep}xy(t)$ and $S_{rep}xy(t)$.
11. Set $k_{rep, xy}(t) = |S_{rep}, xy(t)|$
12. Set $w_{xy}(t) = \sum_{i=1}^{k_{rep, xy}(t)} i(t)$.
13. Set $SCORE_{xy}(t) = p_{table}xy(t) \times w_{xy}(t)$.
14. End
15. /* select sensors for querying */
16. Calculate $S_{xy}(t)$ from $SCORE_{xy}(t)$ and $k_{max}$.
17. /* next time instant */
18. Set $t = t + 1$
19. End

Fig. 13: Pseudocode of the Localization Algorithm

**IV. SIMULATION RESULT**

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.

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 $S = 0.5$, $\theta = 0.5$, 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 $c_{th}$.
Fig. 19: A Trace of Virtual Moves made by the Sensor

C. 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 only 25 seconds for 30 iterations. For Case Study 2, the VFA algorithm took only 3 minutes to complete 50 iterations.

V. 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.

REFERENCES