Movement-Assisted Sensor Deployment

Guiling Wang, Guohong Cao, and Tom La Porta
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
Email: {guiwang,gcao,tlp}@cse.psu.edu

Abstract— **Sensor deployment is an important issue in designing sensor networks. In this paper, we design and evaluate distributed self-deployment protocols for mobile sensors. After discovering a coverage hole, the proposed protocols calculate the target positions of the sensors where they should move. We use Voronoi diagrams to discover the coverage holes and design three movement-assisted sensor deployment protocols, VEC (VECtor-based), VOR (VORonoi-based), and Minimax based on the principle of moving sensors from densely deployed areas to sparsely deployed areas. Simulation results show that our protocols can provide high coverage within a short deploying time and limited movement.**

Index Terms: Sensor deployment, simulations, mobile sensor, Voronoi diagram.

I. Introduction

Wireless sensor networks are expected to be intensively utilized in the future since they can greatly enhance our capability of monitoring and controlling the physical environment. Sensor networks are revolutionizing the traditional methods of data collection, bridging the gap between the physical world and the virtual information world [1], [2], [3], [4]. Due to the inextricable relation with the physical world, the proper deployment of sensors is very important for the successful completion of the sensing tasks issued [5], [6], [7].

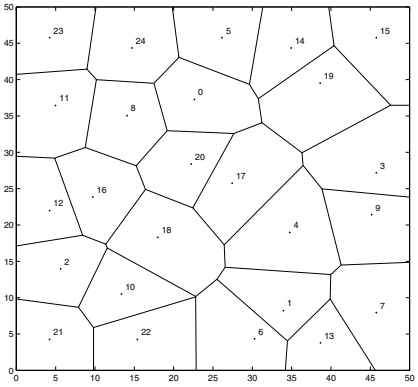
Sensor deployment has received considerable attention recently. Most of these work [5], [8], [9], [10] assume that the environment is sufficiently known and under control. However, when the environment is unknown or hostile such as remote harsh fields, disaster areas and toxic urban regions, sensor deployment cannot be performed manually. To scatter sensors by aircraft is one possible solution. However, using this technique, the actual landing position cannot be controlled due to the existence of wind and obstacles such as trees and buildings. Consequently, the coverage may be inferior to the application requirements no matter how many sensors are dropped. Moreover, in many cases, such as during in-building toxic-leaks detection [11], [12], chemical sensors must be placed inside a building from the entrance of the building. In such cases, it is necessary to make use of mobile sensors, which can move to the correct places to provide the required coverage.

There have been some research efforts on deploying mobile sensors, but most of them are based on centralized approaches. For example, the work in [13] assumes that a powerful cluster head is available to collect the sensor location and determine the target location of the mobile sensors. However, in many sensor deployment environments such as disaster recoveries and battle fields, a central server may not be available and it is hard to organize sensors into clusters due to network partitions. Further, the centralized approach suffers from the problem of single point failure. Sensor deployment has also been addressed in the field of robotics [11], [12], where sensors are deployed one by one, utilizing the location information of previously deployed sensors. This method is not scalable in terms of deployment time and has strong assumptions on the initial placement to guarantee the communication between the deployed and undeployed sensors. In case of network partitions, this method may not be feasible.

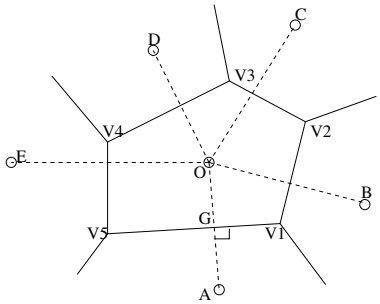
In this paper, we design and evaluate three distributed self-deployment protocols for sensor networks to address the limitations of previous work. Our problem statement is: given the target area, how to maximize the sensor coverage with less time, movement distance and message complexity. Given an area to be monitored, our distributed self-deployment protocols first discover the existence of coverage holes (the area not covered by any sensor) in the target area based on the sensing service required by the application. After discovering a coverage hole, the proposed protocols calculate the target positions of these sensors, where they should move. We use Voronoi diagrams to discover the coverage holes and design three movement-assisted sensor deployment protocols, VEC (VECtor-based), VOR (VORonoi-based), and Minimax based on the principle of moving sensors from densely deployed areas to sparsely deployed areas. By intensive simulations, we evaluate our protocols from various aspects: coverage, deployment time, moving distance, scalability to initial deployment and communication range, etc, and show that our protocols are very effective in terms of coverage, deployment time, and moving distance.

The rest of the paper is organized as follows. Section II introduces the basic knowledge about Voronoi diagram. In section III, we present three self-deployment protocols. Section IV evaluates the performance of the proposed protocols. Based on the simulation results, we justify our design and discuss future work in Section V.

This work was supported in part by the National Science Foundation (CAREER CCR-0092770).



(a) Voronoi diagram



(b) Voronoi polygon $G_p(O)$ of point O

Fig. 1. Voronoi diagram

II. Technical Preliminary: Voronoi Diagram

The Voronoi diagram [14], [15] is an important data structure in computational geometry. It represents the proximity information about a set of geometric nodes. The Voronoi diagram of a collection of nodes partitions the space into polygons. Every point in a given polygon is closer to the node in this polygon than to any other node. Fig. 1(a) is an example of the Voronoi diagram, and Fig. 1(b) is an example of a Voronoi polygon. We define the Voronoi polygon of O as $G_p(O) = (\mathcal{V}_p(O), E_p(O))$, where $\mathcal{V}_p(O)$ is the set of Voronoi vertices of O, and $E_p(O)$ is the set of Voronoi edges. As shown in Fig. 1(b), $\mathcal{V}_p(O) = \{V_1, V_2, V_3, V_4, V_5\}$, $E_p(O) = \{V_1V_2, V_2V_3, V_3V_4, V_4V_5, V_5V_1\}$, $\mathcal{N}(O) = \{A, B, C, D, E\}$, where $\mathcal{N}(O)$ denotes the set of Voronoi neighbors of O. The Voronoi edges of O are the vertical bisectors of the line passing O and its Voronoi neighbors, e.g., V_5V_1 is OA's bisector. All the points inside $G_p(O)$ are closer to O than to any other nodes.

Our sensor deployment protocols are based on Voronoi diagrams. As shown in Fig. 1, each sensor, represented by a number, is enclosed by a Voronoi polygon. These polygons together cover the target field. The points inside one polygon are closer to the sensor inside this polygon than the sensors positioned elsewhere. Thus, if this sensor cannot detect the expected phenomenon, no other sensor can detect it, and then each sensor is responsible for the sensing task in its Voronoi polygon. In this way, each sensor can examine the coverage hole locally, and only needs to monitor a small area around it. To construct the Voronoi polygon, each sensor only needs to know the existence of its Voronoi neighbors, which reduces the communication complexity.

III. Movement-Assisted Sensor Deployment Protocols

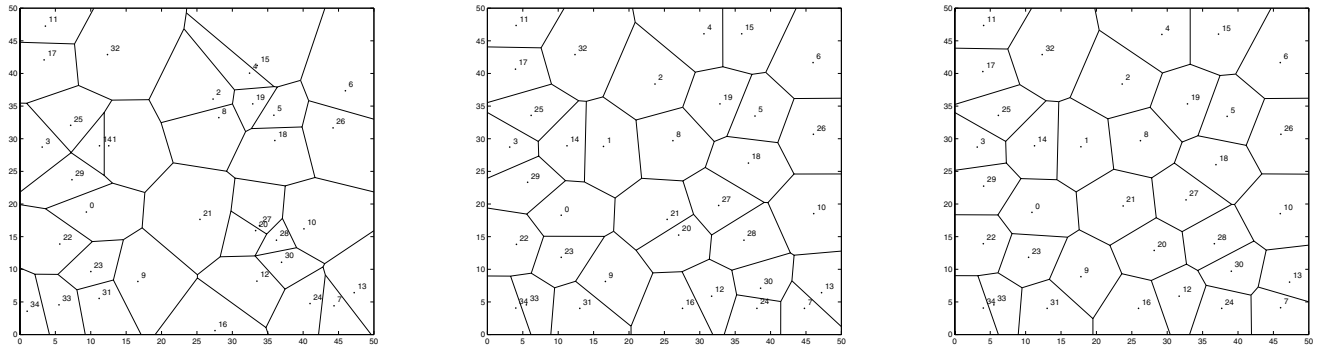
Our sensor deployment protocol runs iteratively until it terminates or reaches the specified maximum round. In each round, sensors first broadcast their locations and construct their local Voronoi polygons based on the received neighbor information. To construct its Voronoi polygon, each sensor first calculates the bisectors of its neighbors and itself based on the location information. These bisectors and the boundary of the target field form several polygons. The smallest polygon encircling the sensor is the Voronoi polygon of this sensor. After the Voronoi polygons have been constructed, they are examined to determine the existence of coverage holes. If any coverage hole exists, sensors decide where to move to eliminate or reduce the size of the coverage hole; otherwise, they stay. Next, we present three movement-assisted sensor deployment protocols: VEC (VECtor-based), VOR (VORonoi-based) and Minimax, based on the principle that evenly distributed sensors can provide better coverage. For these three protocols, VEC *pushes* sensors away from a densely covered area, VOR *pulls* sensors to the sparsely covered area, and Minimax moves sensors to their local center area.

A. The VECtor-based Algorithm(VEC)

VEC is motivated by the attributes of electro-magnetic particles: when two electro-magnetic particles are too close to each other, an expelling force pushes them apart. Assume $d(s_i, s_j)$ is the distance between sensor s_i and sensor s_j. d_{ave} is the average distance between two sensors when the sensors are evenly distributed in the target area, which can be calculated beforehand since the target area and the number of sensors to be deployed are known. The virtual force between two sensors s_i and s_j will push them to move $(d_{ave} - d(s_i, s_j))/2$ away from each other. In case one sensor covers its Voronoi polygon completely and should not move, the other sensor will be pushed $d_{ave} - d(s_i, s_j)$ away. In summary, the virtual force will push the sensors d_{ave} away from each other if coverage hole exists in either of their Voronoi polygons. The virtual force exerted by s_j on s_i is denoted as $\vec{F}_{ij}$, with the direction from s_j to s_i.

In addition to the virtual forces generated by sensors, the field boundary also exert forces, denoted as $\vec{F}_b$, to push sensors too close to the boundary inside. $\vec{F}_b$ exerted on s_i will push it to move $d_{ave}/2 - d_b(s_i)$, where $d_b(s_i)$ is the distance of s_i to the boundary. Since d_{ave} is the average distance between sensors, $d_{ave}/2$ is the distance from the boundary to the sensors closest to it when sensors are evenly distributed.

The final overall force on sensors is the vector summation of virtual forces from the boundary and all Voronoi neighbors. These virtual forces will push sensors from the densely covered area to the sparsely covered area. Thus, VEC is a



(a) Round 0 (b) Round 1 (c) Round 2

Fig. 2. Snapshot of the execution of VEC

Notations:
$\mathcal{N}(s_i)$, $G_p(s_i)$, $\vec{F}_{ij}$, $\vec{F}_b$: defined before
c_i: whether $G_p(s_i)$ is completely covered
$\vec{v}_i$: moving vector of s_i

(1) *Upon entering Discovery phase:*
  (1.1) set $timer$ to be $discovery\_interval$
    enter *Moving phase* upon timeout
  (1.2) broadcast $hello$ after a random time slot

(2) *Upon entering Moving phase:*
  (2.1) set $timer$ to be $moving\_interval$,
    enter *Discovery phase* upon timeout
  (2.2) **if** $c_i = false$ **then**
    call $VEC()$ /* call VOR and Minimax
      in other protocols */
  (2.3) Done when satisfying stop criteria

(3) *Upon receiving $hello$ message from sensor s_j:*
  (3.1) Update $\mathcal{N}(s_i)$ and $G_p(s_i)$
  (3.2) **if** $G_p(s_i)$ is newly covered **then**
    set $c_i = true$
    Broadcast OK /* only for VEC */

/* The following is only for VEC,
  and will be replaced in VOR and Minimax */

(4) *Upon receiving OK message from sensor s_j:*
  (4.1) set c_j = $true$

(5) $VEC()$
  (5.1) $\vec{v_i} = \vec{0}$
  (5.2) **for** each s_j in $\mathcal{N}(s_i)$
    if $(c_j \neq true) \wedge (d_{ave} > d(s_i, s_j))$ **then**
      $|\vec{F_{ij}}| = (d_{ave} - d(s_i, s_j))/2$; $\vec{v_i} = \vec{v_i} + \vec{F_{ij}}$
    if $(c_j = true) \wedge (d_{ave} > d(s_i, s_j))$ **then**
      $|\vec{F_{ij}}| = d_{ave} - d(s_i, s_j)$; $\vec{v_i} = \vec{v_i} + \vec{F_{ij}}$
  (5.3) **if** $(d_{ave}/2 > d_b(s_i))$ **then**
    $|\vec{F_b}| = d_{ave}/2 - d_b(s_i)$; $\vec{v_i} = \vec{v_i} + \vec{F_b}$
  (5.4) **do** *movement adjustment*

Fig. 3. The VEC protocol at sensor s_i

"proactive" algorithm, which tries to relocate sensors to be evenly distributed.

As an enhancement, we add a *movement-adjustment* scheme to reduce the error of *virtual-force*. When a sensor determines its target location, it checks whether the local coverage will be increased by its movement. The local coverage is defined as the coverage of the local Voronoi polygon and can be calculated by the intersection of the polygon and the sensing circle. If the local coverage is not increased, the sensor should not move to the target location. Although the general direction of the movement is correct, the local coverage may not be increased because the target location is too far away. To address this problem, the sensor will choose the midpoint between its target location and its current location as its new target location. If the local coverage is increased at the new target location, the sensor will move; otherwise, it will stay.

Fig. 2 shows how VEC works. Round 0 is the initial random deployment of 35 sensors in a $50m$ by $50m$ flat space, with the sensing range of 6 meters. The initial coverage is 75.7%. After Round 1 and Round 2, the coverage is improved to 92.2% and 94.7% respectively. A formal description of the VEC algorithm is shown in Fig. 3.

B. The VORonoi-based Algorithm (VOR)

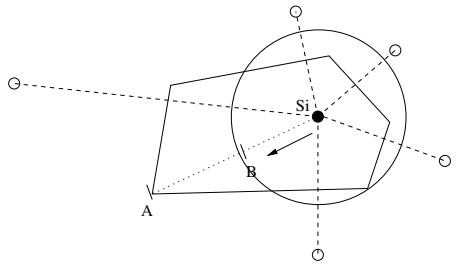
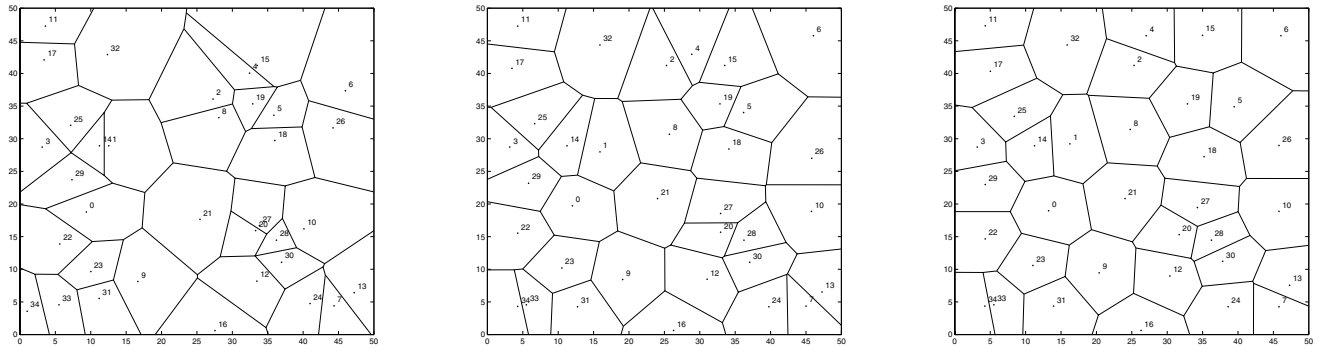


Fig. 5. VOR

Compared to the VEC algorithm, VOR is a *pull-based* algorithm which pulls sensors to their local maximum coverage holes. In VOR, if a sensor detects the existence of coverage holes, it will move toward its farthest Voronoi vertex (denoted as V_{far}). Fig. 5 shows how VOR works. The solid polygon



(a) Round 0 (b) Round 1 (c) Round 2

Fig. 4. Snapshot of the execution of VOR

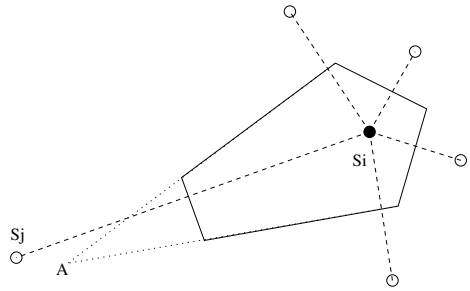


Fig. 6. Inaccurate Voronoi polygon

is $G_p(s_i)$. The small white circles represent s_i's Voronoi neighbors and the large circle represents the sensing circle. Point A is the farthest Voronoi vertex of s_i and $d(A, s_i)$ is longer than the *sensing range*. Sensor s_i moves along line s_iA to Point B, where $d(A, B)$ is equal to the *sensing range*.

We limit the maximum moving distance to be at most half of the communication range to avoid the situation shown in Fig. 6, where s_i is not aware of the existence of s_j because of communication limitations, and its local view of $G_p(s_i)$ (shown in the dotted line) is not correct (shown in the solid line). Otherwise, if s_i moves toward point A and stops at a distance $d(A, B)$ (sensing range), s_i has moved more than needed.

VOR is a greedy algorithm which tries to fix the largest hole. Moving oscillations may occur if new holes are generated due to sensor's leaving. To deal with this problem, we add *oscillation control* which does not allow sensors to move backward immediately. Each time a sensor wants to move, it first checks whether its moving direction is opposite to that in the previous round. If yes, it stops for one round. In addition, the movement adjustment mentioned in VEC is also applied here.

The deployment protocol using VOR is similar to the VEC Protocol, except that in line (2.2) $VEC()$ is replaced by $VOR()$, which is shown below.

Notations:
d_{max}: maximum moving distance
$\vec{v}_{i,f}$: vector from s_i to V_{far}

$VOR()$
(1) $\vec{v_i} = \vec{v}_{i,f}$ - sensing range
(2) shrink $|\vec{v_i}|$ to be d_{max} if $|\vec{v_i}| > d_{max}$
(3) **do** *oscillation control*
(4) **do** *movement-adjustment*

Fig. 4 shows of how VOR works. With the original coverage 75.7%, after round 1 and round 2, the coverage is improved to 89.2% and 95.6% respectively.

C. The Minimax Algorithm

Similar to VOR, Minimax fixes holes by moving closer to the farthest Voronoi vertex, but it does not move as far as VOR to avoid the situation that the vertex which was originally close becomes a new farthest vertex. Minimax chooses the target location as the point inside the Voronoi polygon whose distance to the farthest Voronoi vertex (V_{far}) is minimized. We call this point the *Minimax point*, denoted as O_m. This algorithm is based on the belief that a sensor should not be too far away from any of its Voronoi vertices when the sensors are evenly distributed. Minimax can reduce the variance of the distances to the Voronoi Vertices, resulting in a more regular shaped Voronoi polygon, which better utilizes sensor's sensing circle. Compared with VOR, Minimax considers more information and it is more conservative. Compared with VEC, Minimax is "reactive"; it fixes the hole more directly by moving toward the farthest Voronoi vertex.

The following terms are used to calculate the Minimax point. A circle centered at point O with radius r is denoted $C(O, r)$. The circumcircle of three points V_u, V_v, V_w is denoted as $C(V_u, V_v, V_w)$. Specifically, we define the *Minimax circle* $C_m(O_m, r_m)$ as follows:

Definition 1: Minimax circle $C_m(O_m, r_m)$ is the circle centered at the Minimax point O_m, with radius $r_m = d(O_m, V_{far})$.

The Minimax circle must pass at least two Voronoi vertices, which is proved as the following,

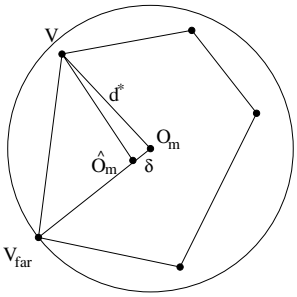


Fig. 7. Proof of Lemma 1

Lemma 1: $C_m(O_m, r_m)$ must pass at least two Voronoi vertices.

Proof: (by contradiction) By definition, $C_m(O_m, r_m)$ must pass V_{far}. Suppose $C_m(O_m, r_m)$ does not pass any other vertices. Let

$$d^* = \max_{V \in \mathcal{V}_p, V \neq V_{far}} \{d(O_m, V)\}. \tag{1}$$

Then $d^* < r_m = d(O_m, V_{far})$. Let $\delta = (r_m - d^*)/2$ and $\hat{O}_m$ be the point on line $O_m V_{far}$ such that $d(O_m, \hat{O}_m) = \delta$ (shown in Fig. 7). Then,

$$d(\hat{O}_m, V_{far}) = r_m - \delta < r_m. \tag{2}$$

Based on *triangle inequality*, we have

$$d(\hat{O}_m, V) \leq d(\hat{O}_m, O_m) + d(O_m, V) = \delta + d(O_m, V), \quad (\forall V \in \mathcal{V}_p \wedge V \neq V_{far}) \tag{3}$$

With (1) and (3), we get

$$d(\hat{O}_m, V) \leq \delta + d^* = r_m/2 + d^*/2 < r_m, \quad (\forall V \in \mathcal{V}_p \wedge V \neq V_{far}) \tag{4}$$

With (2) and (4), we get

$$\max_{V \in \mathcal{V}_p} \{d(\hat{O}_m, V)\} < r_m, \tag{5}$$

which contradicts with the assumption that O_m is the minimax point. ■

Definition 2: The circumcircle of two point V_u and V_w ($C(V_u, V_w)$) is defined as the circle whose center is the mid-point of V_u and V_w and whose radius is $d(V_u, V_w)/2$.

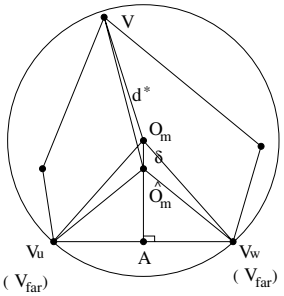


Fig. 8. Proof of Lemma 2

Lemma 2: If $C_m(O_m, r_m)$ passes exactly two Voronoi vertices: V_u and V_w, then $C_m(O_m, r_m) = C(V_u, V_w)$. In other words, if the Minimax circle passes only two Voronoi vertices, it must be centered at the midpoint of these two vertices.

Proof: (by contradiction) Suppose $C_m(O_m, r_m) \neq C(V_u, V_w)$, which means C_m is not the mid-point of V_u and V_w. Let

$$d^* = \max_{V \in \mathcal{V}_p, V \neq V_{far}} \{d(O_m, V)\}, \tag{6}$$

and let $\delta = (r_m - d^*)/2$. Suppose A is the mid-point of V_u and V_w (shown in Fig. 8). Let $\hat{O}_m$ be the point on $O_m A$ such that $d(\hat{O}_m, O_m) = \delta$. Since $\angle O_m \hat{O}_m V_u = \angle O_m \hat{O}_m V_w > \angle O_m A V_u = \angle O_m A V_w = \pi/2$,

$$\begin{cases} d(\hat{O}_m, V_u) < d(O_m, V_u) = r_m. \\ d(\hat{O}_m, V_w) < d(O_m, V_w) = r_m. \end{cases} \tag{7}$$

With *triangle inequality*, we have

$$d(\hat{O}_m, V) \leq d(\hat{O}_m, O_m) + d(O_m, V) = \delta + d(O_m, V), \quad (\forall V \in \mathcal{V}_p \wedge V \neq V_{far}) \tag{8}$$

With (6) and (8), we get

$$d(\hat{O}_m, V) \leq \delta + d^* = r_m/2 + d^*/2 < r_m, \quad (\forall V \in \mathcal{V}_p \wedge V \neq V_{far}) \tag{9}$$

With (7) and (9), we get

$$\max_{V \in \mathcal{V}_p} \{d(\hat{O}_m, V)\} < r_m, \tag{10}$$

which contradicts with the assumption that O_m is the minimax point. ■

If the Minimax circle passes more than two Voronoi vertices, it is the circumcircle of these vertices. To find the Minimax point, we only need to find all the circumcircles of any two and any three Voronoi vertices. Among those circles, the one with the minimum radius covering all the vertices is the Minimax circle. The center of this circle is the Minimax point. We formally state this claim and prove it in the following:

Theorem 1: Let $\Gamma = \{C(V_u, V_v) \mid C(V_u, V_v)$ covers all vertices in $\mathcal{V}_p$, and $V_u, V_v \in \mathcal{V}_p\} \bigcup \{C(V_u, V_v, V_w) \mid C(V_u, V_v, V_w)$ covers all vertices in $\mathcal{V}_p$, and $V_u, V_v, V_w \in \mathcal{V}_p\}$, then $C_m(O_m, r_m) \in \Gamma$ and $\forall C(O, r) \in \Gamma, r \geq r_m$.

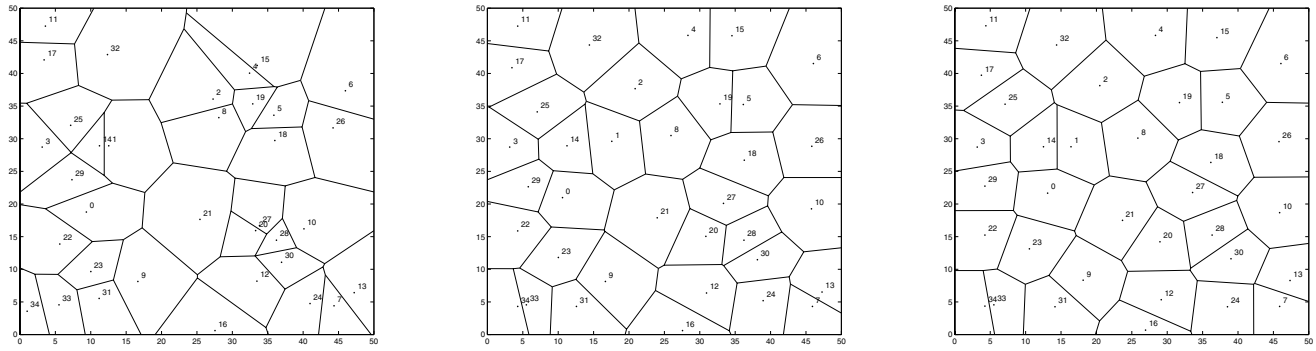
Proof: By Lemma 1, $C_m(O_m, r_m)$ passes at least two Voronoi vertices.

Case1: $C_m(O_m, r_m)$ passes three or more Voronoi vertices. Then it is the circumcircle of any three vertices it passes. With the definition of Γ, $C_m(O_m, r_m) \in \Gamma$. Based on the definition of $C_m(O_m, r_m)$, $r_m = \min_{C(V,r) \in \Gamma} \{r\}$.

Case2: $C_m(O_m, r_m)$ passes exactly two Voronoi vertices: V_u and V_w. Based on Lemma 2, $C_m(O_m, r_m) = C(V_u, V_w)$. With the definition of Γ, $C_m(O_m, r_m) \in \Gamma$. Based on the definition of $C_m(O_m, r_m)$, $r_m = \min_{C(V,r) \in \Gamma} \{r\}$.

■

Based on Theorem 1, we obtain the algorithm to calculate the Minimax point which is formally stated as below. The complexity of this algorithm is $O(n^3)$, where n is the number



(a) Round 0 (b) Round 2 (c) Round 3

Fig. 9. Snapshot of the execution of Minimax

of the Voronoi vertices. The computational cost is not high since a Voronoi polygon usually has only a few vertices.

$Minimax()$:
(1) Initialize $n = |\mathcal{V}_p|$
(2) **for** u = 1, 2, ..., $n-2$
  for v = $u+1$, $u+2$, ..., $n-1$
    for w = $v+1$, $v+2$, ..., n
      Calculate $C(V_u, V_v, V_w)$
      if V is inside $C(V_u, V_v, V_w)$ $\forall V \in \mathcal{V}_p$ **then**
        Record it.
(3) **for** u = 1, 2, ..., $n-1$
  for v = $u+1$, $u+2$, ..., n
    Calculate $C(V_u, V_v)$
    if V is inside $C(V_u, V_v)$ $\forall V \in \mathcal{V}_p$ **then**
      Record it.
(4) choose one with minimum radius and
  set the target location to be the center
(5) **do** *movement-adjustment*

Fig. 9 shows how Minimax works. With the original coverage 75.7%, after round 1 and round 2, the coverage is improved to 92.7% and 96.5% respectively.

D. Termination

The algorithm terminates naturally based on the *movement-adjustment* heuristic (explained in Section III-A), which does not allow sensors to move unless the local coverage can be increased. The total coverage, bounded by 100%, increases as the local coverage increases. Based on the attributes of Voronoi diagram, the local coverage increase of one sensor does not affect the local coverage of another sensor. Thus, sensors will stop naturally when the best coverage is obtained.

In some applications, the coverage requirement is not that high, and it is not efficient to move sensors to get a very small coverage increase. In this case, it may be necessary to terminate the deployment process before the maximum coverage is reached to save power and reduce the deployment time. To terminate the deployment procedure earlier, we use a threshold ϵ, defined as the minimum increase in coverage below which a sensor will not move. With a larger ϵ, the deployment will finish earlier. When $\epsilon = 0$, sensors stop when the best coverage is obtained.

E. Optimizations

In some cases, the initial deployment of sensors may form clusters, as shown in Fig. 10, resulting in low initial coverage. In this case, sensors located inside the clusters can not move for several rounds, since their Voronoi polygons are well covered initially. This problem prolongs the deployment time, which is shown in Fig. 10, where some sensors are still clustered together after the sixth round. To reduce the deployment time in this situation, we propose an optimization which detects whether too many sensors are clustered in a small area. The algorithm "explodes" the cluster to scatter the sensors apart, if necessary, which works as follows. Each sensor compares its current neighbor number to the neighbor number it will have if sensors are evenly distributed. If a sensor finds the ratio of these two numbers is larger than a threshold, it concludes that it is inside a cluster and chooses a random position within an area centered at itself which will contain the same number of sensors as its current neighbors in the even distribution. The explosion algorithm only runs in the first round. It scatters the clustered sensors and changes the deployment to be close to random.

IV. Performance Evaluations

A. Objectives, Metrics, and Methodology

We measure the performance of the proposed protocols from two aspects: *deployment quality* and cost. *Deployment quality* is measured by the sensor coverage and the time to reach this coverage. Deployment time is determined by the number of rounds needed and the time of each round. The duration of each round is primarily determined by the moving speed of sensors, which is the mechanical attribute of sensors. Thus, we only use the number of rounds to measure the deployment time. The *Cost* has two components. One is the sensor cost; to reach a certain coverage, the proposed protocol needs fewer sensors than random deployment of static sensors. The other is the energy consumption of the deployment. Both mechanical movement and electronic communication consume energy, of

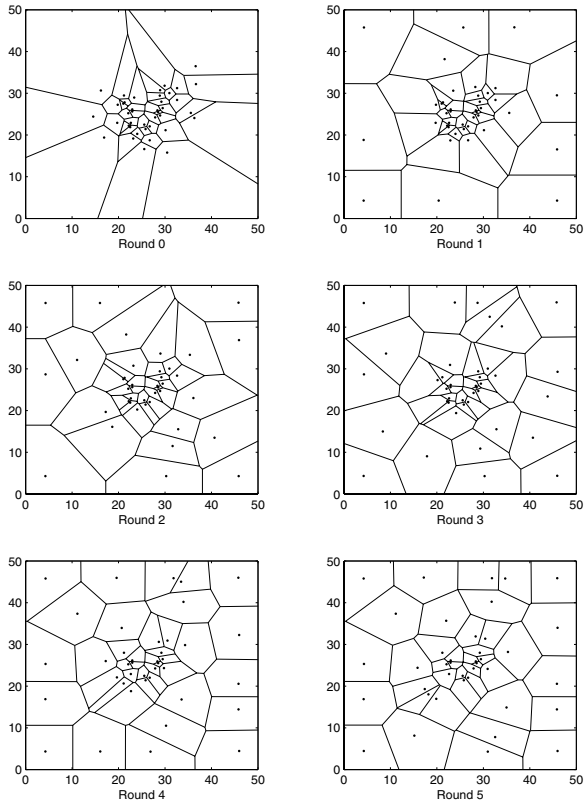


Fig. 10. Working procedure (VOR)

which mechanical motion is the major part. Hence we choose the moving distance as the evaluation metric.

We measure the sensor coverage and moving distance under various system parameters: sensor density, field size, topology, communication range, and ϵ. With certain sensing range, the sensor density determines the sensor coverage that can be reached, and the difficulty to reach it. We choose 30, 35, 40, and 45 sensors per $50m * 50m$ field as the sensor density in our simulation. We also vary the field size ($50m * 50m$ to $150m * 150m$) and the number of sensors to test the protocols' extendibility to large scenarios. We consider two kinds of initial deployments. One is the random distribution which can be used to model many cases such as when the sensors are dropped by an airplane from a high altitude. The other is the normal distribution, which can be used to model the case where sensors form a cluster. By varying the standard deviation, we can control the dense degree of the sensor clustering.

Communication range is another important factor since it affects the accuracy of the constructed Voronoi diagram depending on which sensors detect the coverage hole and choose the target locations. We will vary the communication range from 10m to 28m to see how the performance is affected. $20m$ is what is used in most sensor prototypes. We choose higher communication range to quantify the performance improvement. $10m$ is the point at which network partitions become common. For example, when 36 sensors are deployed in a $50m$ by $50m$ field and when they are in their optimal sensing position, the minimum distance between two sensors is $6 * \sqrt{(2)}$, which is about $8.5m$. If the communication range is less than this value, the network is totally disconnected when sensors are in their optimal sensing position.

We implemented the proposed protocols in ns2 (version 2.1b9a). The target field is chosen to be $50m * 50m$ flat space except in the evaluation of our scheme's sensitivity to the field size (in section IV-B.3). The initial placement of sensors follows random distribution except in the evaluation of the impact of topology. All simulation results are under $\epsilon = 0$ except in the testing of our algorithm's sensitivity to ϵ. We use 802.11 as the MAC layer protocol and DSDV as the routing protocol. The physical layer is modeled after the RF MOTE from Berkeley, with 916.5MHZ OOK 5kbps as the bandwidth and 20 meters as the transmission range except in the evaluation of the impact of communication range. Based on the information from [16], we set the *sensing range* to be 6 meters. This is consistent with other current sensor prototypes, such as Smart Dust (U.C.Berkeley), CTOS dust, Wins (Rockwell)[17], and JPL[18].

B. Simulation Results

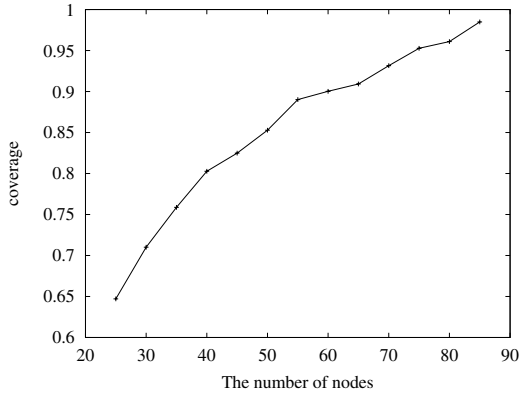
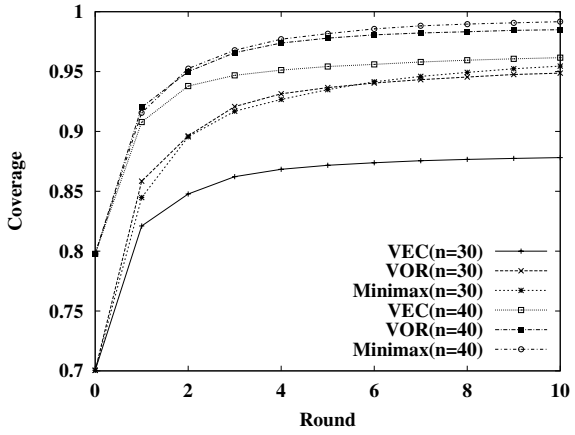


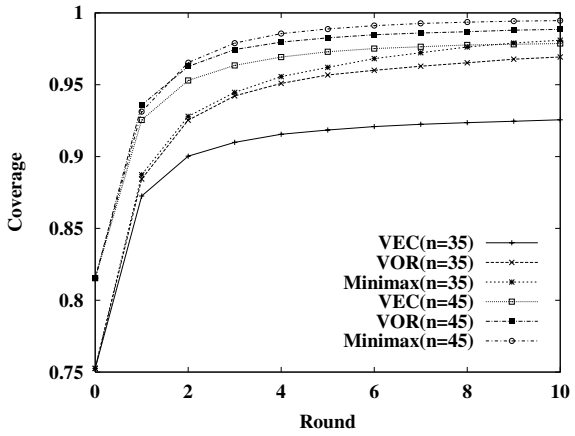
Fig. 11. Coverage (randomly deployed)

1) Coverage and Sensor Cost: The coverage of randomly deployed sensor networks under different sensor density is shown in Fig. 11, which shows that about 85 sensors are required to reach 98% coverage. In contrast, by using our sensor deployment protocols, only 40 sensors (shown in Fig. 12(a)) are needed to reach the same coverage.

From Fig. 12(a) and Fig. 12(b), we can see Minimax performs the best, while VEC performs the worst. Minimax fully utilizes the local Voronoi polygon. It fixes the coverage hole directly by moving toward the largest hole, while avoiding the possible negative effect by the sensor's movement. In addition, by locating sensors in the Minimax point, Minimax lowers the variance of a sensor's distances to its Voronoi vertices, resulting in a more regular shaped Voronoi polygon. VOR fixes the coverage hole more greedily, but lacks a comprehensive consideration. It is expectable that VOR performs worse than Minimax. One thing to be noted is in the first round, VOR is better than Minimax. This is because initially the coverage

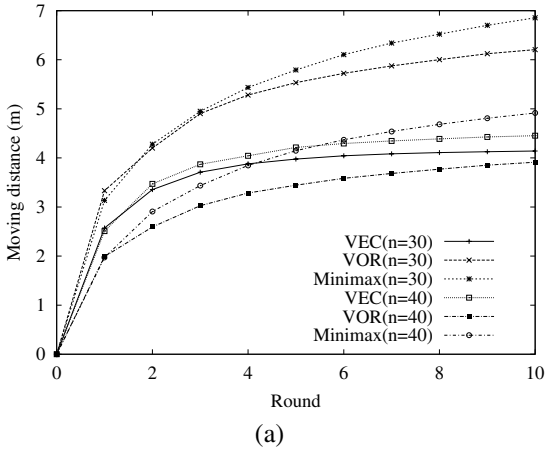


(a)

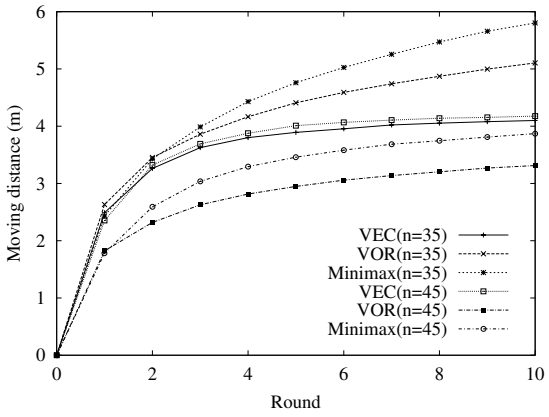


(b)

Fig. 12. Coverage



(a)



(b)

Fig. 13. Moving distance

holes are normally bigger than in the middle of the deployment procedure, thus to fix them greedily can result in a higher coverage increase.

VEC performs the worst for several reasons. The primary one is that VEC is sensitive to the initial deployment. Consider an extreme situation, where sensors are located in the same line with the same inter-distance. In this case, no sensor will move, since the virtual forces offset each other, though there are large coverage holes. If the sensors are located in a similar relative position initially, VEC does not perform well. In addition, VEC neither considers coverage holes nor utilizes any geometric information from the Voronoi polygon when choosing the target location. It tries to reach a relatively balanced position among the sensors, which is very hard, for the difficulty of obtaining an exact global even distribution from only local information.

2) Moving Distance: Fig. 13 shows the average cumulative moving distance after each round. From the figure, we can see an interesting phenomena about VEC: the moving distance is similar under different sensor densities. This is because VEC fixes coverage holes by pushing sensors into a relatively even distribution. In VEC, sensors are pushed by the virtual forces, which are determined by the difference between the average distance of sensors when they are evenly distributed and the individual inter-distances. Both values increase with a low density and decrease with a higher density. Thus, the difference is not that sensitive to the sensor density. In contrary, Minimax and VOR relocate sensors by measuring the coverage holes, which are larger under lower density and smaller under high density. In addition, the curves of VEC is the flattest among these three since the movement is very small when sensors arrive at a relatively balanced position.

Between Minimax and VOR, the former always moves a longer distance. Minimax not only tries to fix holes, but also tries to reach more regular shaped Voronoi polygons. Thus, after the first two rounds and the remaining holes are relatively small, VOR moves sensors slightly while Minimax makes sensor move longer to the Minimax points.

3) Extendibility to Large Scenarios: To test the sensitivity of our methods to field size and network scale, we fix the sensor density to be 40 sensors per $50m * 50m$ field, and vary the field size from $50m * 50m$ to $150m * 150m$. The coverage reached and the moving distance after ten rounds is shown in Table I. From the data, we can see that our algorithms are extensible to large deployment scenarios, because the communication and movement are kept local in our protocols.

In addition, we notice that the performance is slightly better with large field size. This is because the sensing circle is better utilized inside the field than beside the boundary. With a large field, the percentage of sensors inside is higher.

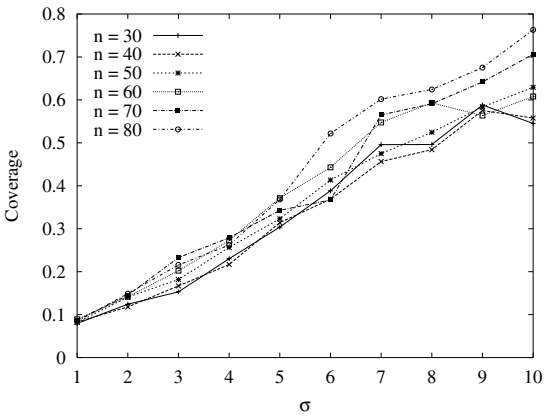


Fig. 14. Reference normal distribution

4) Impact of Topology: The simulation results in the previous sections indicate that our algorithms perform quite well if the initial deployment is random. In this section, we examine the performance on other types of topologies. Fig. 14 shows the coverage under different σ and sensor density when sensors are deployed following the normal distribution. With as high as 80 sensors, the coverage cannot reach 80% even when σ is 10. By comparing this figure with Fig. 15, we can see the effectiveness of our algorithms.

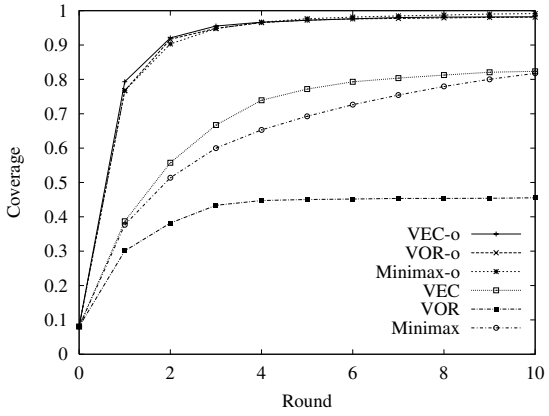
Fig. 15 presents a dense clustering scenario. All 40 sensors are deployed around the center of the field with a normal distribution. We use $\sigma = 1$ and $\sigma = 5$ to represent the clustering degree of sensors. It is a very rigid situation when σ is equal to 1, since the initial coverage is below 10%, and about 28 sensors are located within the circle of one meter radius. The data includes both the basic version of the algorithm and the optimized version. In Fig. 15, VEC-o, VOR-o, and Minimax-o represent the optimized version.

	R	**C**	**D** (m)	**E**	**M**
VEC-o	10	98.29%	23.00	1.24	3.43
VEC	20	84.04%	16.00	1.39	4.37
VOR-o	10	98.11%	21.75	1.17	4.22
VOR	20	46.21%	8.19	1.69	1.32
Minimax-o	10	99.15%	22.81	1.20	5.38
Minimax	20	89.11%	13.76	1.18	5.73

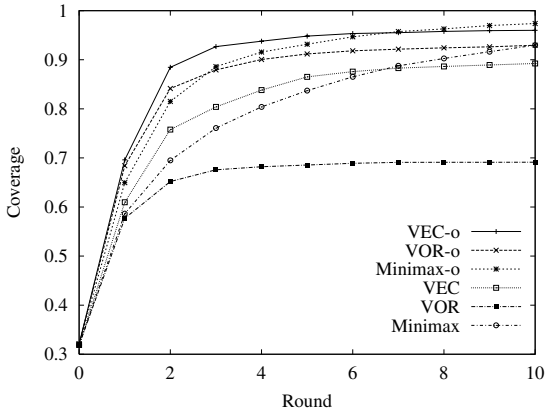
TABLE II

IMPROVEMENT OF THE OPTIMIZATION ($\sigma = 1; n = 40$): **E** *measures the effectiveness of moving, which is the ratio of actual moving distance to the distance between the initial position and the final position.* **M** *shows the average number of movements of sensors.* **C** *and* **D** *refer to the coverage and to the moving distance respectively.* **R** *is the Round number to record these data.*

As mentioned in SectionIII-E, our basic algorithms have difficulties in dealing with this high-degree clustering, espe-



(a) σ=1 (n=40)



(b) σ=5 (n=40)

Fig. 15. Normal Distribution

cially when $\sigma = 1$. In this case, VEC works better than other algorithms in the first several rounds, because it pushes sensors into balanced positions aggressively. Though Minimax can not increase the coverage as quickly as in the random case, it can increase the coverage steadily. It surpasses VEC after the tenth round when $\sigma = 1$, and after the seventh round when $\sigma = 5$. This again demonstrates that Minimax has advantage in coverage.

Though our basic algorithm cannot deal well with the problem of a high degree clustering, the high coverage reached after the first round shows the effectiveness of our optimized algorithm in scattering sensors from being stick together. In addition, the explosion done in the first round does not add additional cost, instead, the deployment cost is reduced compared with the basic version. Table II presents additional data. From the number of movements and the moving efficiency, we can see that the optimized algorithm does not add additional cost while the round number and reached coverage tell us the optimized algorithm reduces the deployment time significantly.

5) Impact of Communication Range: We randomly deploy 40 sensors in the platform and vary the transmission range from $10m$ to $28m$ to evaluate its impact on the performance. Fig. 16 and Fig. 17 show the moving distance and the coverage reached after 10 rounds under different communication range.

	$50m * 50m$		$100m * 100m$		$150m * 150m$	
	Coverage	Moving Distance(m)	Coverage	Moving Distance(m)	Coverage	Moving Distance(m)
VEC	97.15%	4.45	97.41%	4.38	97.52%	4.33
VOR	98.50%	3.91	98.73%	3.78	98.80%	3.75
Minimax	99.17%	4.91	99.35%	4.79	99.67%	4.82

TABLE I

COVERAGE REACHED UNDER DIFFERENT FIELD SIZE

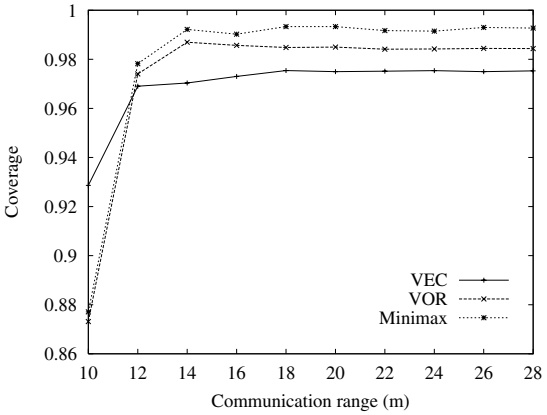


Fig. 16. Impact on coverage (n = 40)

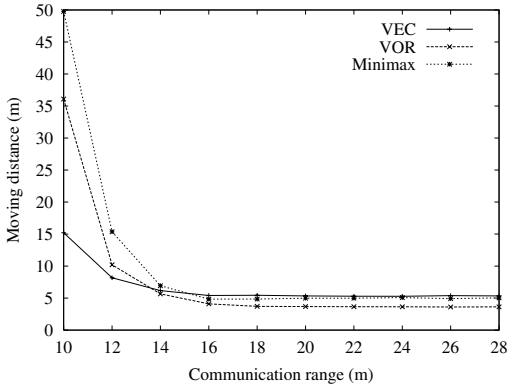


Fig. 17. Impact on moving distance (n = 40)

As can be seen, when the communication range is lower than $12m$, the performance of our algorithms is reduced. When the communication range is too low, most sensors do not know all their Voronoi neighbors, and the constructed Voronoi diagram is not very accurate. Consequently, sensors may get some false coverage holes and make wrong decisions about the target location. Among these three protocols, VEC is affected the least by the low communication range in terms of both moving distance and coverage, since it only utilizes the Voronoi polygon to determine whether to move or not, but not the target position. Sensors are only affected by the virtual force from their neighbors within d_{ave}, so lack of knowledge of sensors far away does not affect the accuracy of the calculated target location.

Fig. 16 and Fig. 17 show that when the communication range is greater than $12m$, the performance is quite good. However, our trace file shows that even when the communication range is $26m$, some constructed Voronoi diagrams are not accurate. This indicates that the heuristics used to deal with the inaccuracy of constructing Voronoi polygons are very effective.

6) Coverage increase threshold ϵ: In this section, we evaluate how ϵ affects the performance. In Table III, the values shown in the leftmost column are the product of ϵ used by sensors for determining whether to move and the number of sensors. As can be seen, with a smaller ϵ, a higher coverage can be reached, while the deployment cost is also increased. By properly setting this threshold, we can save time and energy by trading off a very small coverage (less than 1% in the table). From table III, we also find that, when $\epsilon * n > 0.5\%$, each sensor only moves about 20% more than the direct distance from the starting point to the destination. This also means that our protocols only incur an approximately additional 20% movement overhead compared to the centralized approach, which may not be feasible and suffer from single point failure problem.

V. DISCUSSION AND FUTURE WORK

This paper addressed the problem of placing sensors in a target field to maximize the sensing coverage. Based on Voronoi diagrams, we designed three distributed protocols to move mobile sensors from densely deployed areas to sparsely deployed areas in an iterative way. Simulation results verified the effectiveness of our protocols and provided a baseline for performance under ideal conditions. In this section, we discuss some open issues that will be addressed in the future.

A. Optimal Movement vs Communication

Our protocols require sensors to move iteratively, eventually reaching the final destination. Other approaches can be envisioned in which the sensors move only once to their destination to minimize the sensor movement. Two such approaches are a centralized approach and an approach using simulated movement. Our results show that our distributed algorithms only incur an approximately additional 20% movement overhead, compared to these algorithms. However, we provide significant benefit in other dimensions as described below.

Although the *centralized* approach may minimize the sensor movement, a central server architecture may not be feasible in some applications. For example, in the battle field, sensors are responsible for detecting abnormal phenomena and warning soldiers nearby. No central server in the battle field can help these sensors, and an individual sensor does not have the

$\epsilon * n$	VEC					VOR					Minimax				
	R	C(%)	D	M	E	R	C(%)	D	M	E	R	C(%)	D	M	E
0.0%	47.56	98.44	5.18	3.67	1.62	37.88	98.74	4.19	6.30	1.60	41.39	99.49	5.31	6.88	1.50
0.1%	14.23	97.87	4.57	2.55	1.47	12.00	98.57	3.62	2.74	1.47	13.71	99.11	4.28	3.67	1.35
0.5%	8.00	97.24	4.02	1.70	1.41	7.00	97.92	3.13	1.67	1.30	9.44	98.37	3.49	2.11	1.24
1%	6.22	96.62	3.62	1.26	1.33	5.10	97.19	2.85	1.26	1.23	6.78	97.84	3.11	1.60	1.21
2%	5.90	95.87	3.21	1.01	1.23	4.70	96.54	2.55	0.98	1.18	5.90	96.82	2.60	1.18	1.14

TABLE III

IMPACT OF ϵ ($n = 40$) (**R** *is the round number when all sensors stop;* **C**, **D**, **M**, *and* **E** *have been defined in Table II. These values are obtained in the stopping round* **R**)

computation power of a center server. In the case of an in-building toxic-leak, mobile sensors have to self deploy into the building from outside without server support. Further, the centralized approach suffers from the problem of single point failure.

Another alternative method is to let sensors stay fixed and obtain their final destinations by simulated movement. With the same round-by-round procedure, sensors calculate their target locations, logically move there, and exchange these locations with the sensors which would be their neighbors as if they had actually moved. The real movement only happens at the last round when they get the final destinations. We did not deploy this alternative method for two reasons. First, an approach using simulated movement is susceptible to poor performance under network partitions which are likely to occur in a sensor deployment. If a network partition occurs, each partition will exercise the movement algorithms without knowledge of the others. Consequently, the obtained final destination is not accurate and the required coverage cannot be reached. Using real movement, the network partitions will be healed allowing all sensors to be eventually considered in the algorithm. A second reason is the high communication overhead. To guarantee logical neighbors are reached, a network-wide broadcast is needed when using simulated mobility. If this network-wide broadcast is implemented by gossiping, the message complexity is at minimum $2rn^2$. Using actual mobility as in our protocols, a much lower message complexity, $2rn$, is enough.

B. Sensing Area

In this paper, the sensing area of each sensor is assumed to be a disk with radius $6m$. This is the ideal case which provides us with a baseline of the sensor placement problem. In future work, we will address varying sensing ranges and investigate such cases. Here, we discuss these issues.

Our protocols can deal well with the case of a larger or smaller sensing radius if the sensing area is uniformly a disk. The performance of the protocol depends more on the ratio of communication range to sensing range than the absolute sensing range. As the sensing range decreases with regard to the communication range, our protocols will perform very well because they can accurately construct the Voronoi diagrams. As the sensing range increases, we need to enlarge the broadcast hops to better construct the Voronoi cells.

If the sensing area is an irregular shape, instead of a disk, sensors can still check their Voronoi cells to determine the coverage holes. In this case, we can decrease the sensing range used in our protocols to account for the reduced coverage. In future work, we will study our protocol's sensitivity to the sensing area.

C. Extend to Large Sensor Networks

In our simulation, we use a $50m * 50m$ (and $150m * 150m$) field and tens of sensors. In some situations, perhaps thousands of sensors are needed in a very large field. Simulation shows that the protocols presented here are not sensitive to the scale of the network and the target field; the performance of our algorithm depends on the *density* of sensors used in the field. This is because communication and movement are kept local.

REFERENCES

[1] W. R. Heinzelman, J. Kulik and H. Balakrishnan, "Adaptive Protocols for Information Dissemination in Wireless Sensor Network," *Mobicom'99*, August 1999.

[2] C. Intanagonwiwat , R. Govindan and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication," *MobiCOM '00*, August 2000.

[3] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communications of the ACM*, May 2000.

[4] K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie, "Protocols for self-Organization of a Wireless Sensor Network," *IEEE Personal Communication*, October 2000.

[5] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan and K. k. Saluja, "Sensor Deployment Strategy for Target Detection," *WSNA*, 2002.

[6] S. Tilak, N. B. Abu-Ghazaleh and W. Heinzelman, "Infrastructure Tradeoffs for Sensor Networks," *WSNA*, 2002.

[7] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks (Elsevier) Journal*, March 2002.

[8] S. Dhillon, K. Chakrabarty and S. Iyengar, "Sensor placement for grid coverage under imprecise detections," *Proc.International Conference on Information Fusion*, 2002.

[9] S. Meguerdichian , F. Koushanfar, M. Potkonjak and M. B. Srivastava, "Coverage Problems in Wireless Ad-hoc Sensor Network," *IEEE INFOCOM'01*, April 2001.

[10] S. Meguerdichian , F. Koushanfar, G. Qu and M. Potkonjak, "Exposure in Wireless Ad-Hoc Sensor Networks," *Mobicom*, 2001.

[11] A. Howard, M. J. Mataric and G. S. Sukhatme, "Mobile Sensor Networks Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," *the 6th International Symposium on Distributed Autonomous Robotics Systems*, June 2002.

[12] A. Howard, M. J. Mataric and G. S. Sukhatme, "An Incremental Self-Deployment Algorithm for Mobile Sensor Networks," *Autonomous Robots, Special Issue on Intelligent Embedded Systems*, September 2002.

[13] Y. Zou and K. Chakrabarty, "Sensor deployment and target localization based on virtual forces," *INFOCOM*, 2003.

[14] D. Du and F. Hwang S. Fortune, "Voronoi diagrams and Delaunay triangulations," *Euclidean Geometry and Computers*, 1992.

[15] F. Aurenhammer, "Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, 1991.

[16] "http://www-bsac.eecs.berkeley.edu/~shollar," .

[17] "http://wins.rsc.rockwell.com," .

[18] "http://sensorwebs.jpl.nasa.gov," .