

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—Adequate coverage is very important for sensor networks to fulfill the issued sensing tasks. In many working environments, it is necessary to make use of mobile sensors, which can move to the correct places to provide the required coverage. In this paper, we study the problem of placing mobile sensors to get high coverage. Based on Voronoi diagrams, we design two sets of distributed protocols for controlling the movement of sensors, one favoring communication and one favoring movement. In each set of protocols, we use Voronoi diagrams to detect coverage holes and use one of three algorithms to calculate the target locations of sensors if holes exist. Simulation results show the effectiveness of our protocols and give insight on choosing protocols and calculation algorithms under different application requirements and working conditions.

Index Terms—Mobile Sensor Networks, Sensor Coverage, Distributed Algorithm

I. INTRODUCTION

Wireless sensor networks can greatly enhance our capability to monitor and control 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 [11], [15], [24], [28]. Sensor nodes must be deployed appropriately to reach an adequate coverage level for the successful completion of the issued sensing tasks [5], [21].

In many potential working environments, 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 positions cannot be controlled because of 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 [12], [13], chemical sensors must be placed inside a building from the outside. In these scenarios, it is necessary to make use of mobile sensors, which can move to the correct places to provide the required coverage. One example

of a mobile sensor is the Robomote [26]. These sensors are smaller than $0.000047m^3$ and cost less than 150 dollars.

Most previous research efforts on deploying mobile sensors are based on centralized approaches. For example, the work in [30] assumes that a powerful cluster head is available to collect the sensor locations and determine the target locations of the mobile sensors. However, in many sensor deployment environments such as disaster areas and battle fields, a central server may not be available. It may also be hard to organize sensors into clusters due to network partitions. Further, centralized approaches introduce a single point of failure. Sensor deployment has also been addressed in the field of robotics [12], where sensors are deployed iteratively one by one, utilizing the location information obtained from the previous deployment. Since sensors are deployed one by one, the deployment time is very long which can significantly increase the network initialization time.

In this paper, we propose two sets of distributed protocols for controlling the movement of sensors to achieve target coverage: *basic protocols* and *virtual movement protocols*. In the basic protocols, sensors move iteratively, eventually reaching the final destination. In each iteration, sensors detect coverage holes using a Voronoi diagram. If holes exist, they calculate the target locations to heal the holes and move. In the virtual movement protocols, sensors do not perform iterative physical movement. Instead, after calculating the target locations, sensors move virtually, and exchange these new virtual locations with the sensors which would be their neighbors if they had actually moved. The real movement only occurs when the communication cost to reach their logical neighbors is too high or when they determine their final destinations.

In both the basic and virtual movement protocols, three algorithms, VEC, VOR, and Minimax, are proposed to calculate the target locations if coverage holes exist. In VEC, sensors move away from a dense area; in VOR, sensors migrate towards holes; in Minimax, sensors also move towards holes, but more conservatively with the

consideration of not generating new holes. Simulation results show that our distributed protocols are effective in terms of coverage, deployment time and movement.

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

II. PRELIMINARIES

A. Localization Techniques

Location awareness is important for wireless sensor networks since many applications such as environment monitoring and target tracking depend on knowing the locations of sensor nodes. Due to the ad hoc nature of such networks, each node must determine its location through a location discovery process. For outdoor systems, Global Positioning System (GPS) [3] is one method for this purpose. GPS may not be cost effective or work well indoors.

Many techniques have been proposed to enable each node to determine its location indoors with only limited communication with nearby nodes. Most of these methods exploit received signal strength [22], time difference of arrival of two different signals [25], and angle of arrival [7]. Hu et al. [14] have provided detailed discussion of these techniques. In the subsequent discussions of this paper, we assume that sensor nodes know their locations.

B. Path Planning

In systems that exploit mobile sensors, finding paths on which these mobile sensors can move to desiring destinations, especially when there exist obstacles in the field, is an important problem. The problem has been studied in the area of robotics [6], [17]. Recently, Li et al. [18] studied the problem in sensor networks. They combined the above methods to find the best motion path, and modified them to exploit the distributed nature of sensor networks. In this paper, we do not study this problem further; we assume that mobile sensors can move to any location where they are asked to move based on the existing techniques. We comment more on the impact of this assumption in Section V-B.

C. Sensing Model

Each type of sensor has its unique sensing model characterized by its sensing area, resolution and accuracy. The sensing area depends on multiple factors such as the

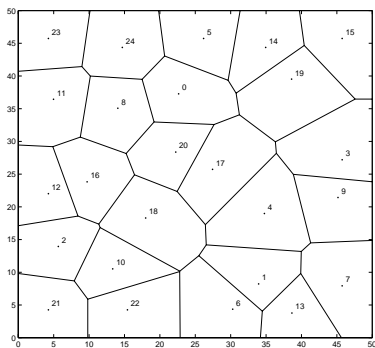
strength of the signals generated at the source, the distance between the source and the sensor, the attenuation rate in propagation, and the desired confidence level of sensing. Let us consider an application [8] in which a network of acoustic sensors is deployed for detecting mobile vehicles. Due to signal attenuation, sensors closer to a vehicle can detect higher strength of acoustic signals than sensors farther away from the vehicle, and thus have higher confidence for detecting the vehicle. Therefore, given a confidence level, we can derive a sensing range surrounding each sensor. In this paper, we only consider the isotropic sensing models. Each sensor node is associated with a sensing area which is represented by a circle with the same radius. This is a common assumption when comparing algorithms for sensing coverage [20], [21].

D. Voronoi Diagram

The Voronoi diagram [4], [9] 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. Figure 1(a) is an example of the Voronoi diagram, and Figure 1(b) is an example of a Voronoi polygon. We define the Voronoi polygon of s_0 as $G_0 = \langle \mathcal{V}_0, \mathcal{E}_0 \rangle$, where \mathcal{V}_0 is the set of Voronoi vertices of s_0 , and \mathcal{E}_0 is the set of Voronoi edges. As shown in Figure 1(b), $\mathcal{V}_0 = \{V_1, V_2, V_3, V_4, V_5\}$, and $\mathcal{E}_0 = \{V_1V_2, V_2V_3, V_3V_4, V_4V_5, V_5V_1\}$. We use \mathcal{N}_0 to denote the set of Voronoi neighbors of s_0 . In Figure 1(b), $\mathcal{N}_0 = \{s_1, s_2, s_3, s_4, s_5\}$. The Voronoi edges of s_0 are the vertical bisectors of the line passing s_0 and its Voronoi neighbors, e.g., V_1V_5 is s_0s_1 's bisector.

Our sensor deployment protocols are based on Voronoi diagrams. As shown in Figure 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. If this sensor cannot detect the expected phenomenon in its Voronoi polygon, no other sensor can detect it. Therefore, to examine coverage holes, each sensor only needs to check its own Voronoi polygon. If its sensing area cannot cover the polygon, there are some coverage holes.

To construct the Voronoi polygon, sensors first calculate the bisectors of their neighbors and themselves. These bisectors (and possibly the boundary of the target field) form several polygons. The smallest polygon encircling the sensor is the Voronoi polygon of this sensor.



(a) Voronoi diagram

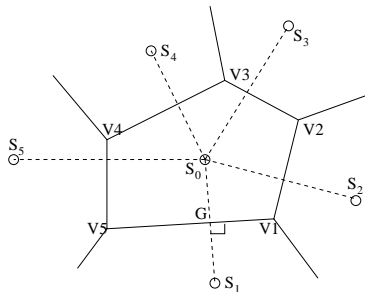
(b) Voronoi polygon G_0 of s_0

Figure 1. Voronoi diagram

E. Sensing Range versus Communication Range

In a distributed case, sensors can exchange the location information by broadcasting. It is possible that some Voronoi neighbors of a sensor are out of its communication range, and consequently, the calculated polygon of this sensor is not accurate. If the sensing range is much shorter than the communication range, then the inaccurate construction of Voronoi cell will not affect the detection of coverage holes. This is because, if Voronoi neighbors cannot reach each other by direct communication, their distance is large enough that there is a coverage hole. If communication range is similar to the sensing range, sensors may mis-detect coverage holes. We describe our heuristics to deal with the inaccurate construction of the Voronoi polygons in Section III.

III. THE BASIC DEPLOYMENT PROTOCOLS

Our deployment protocol runs iteratively. In each round, sensors first broadcast their locations and construct their Voronoi polygons based on the received neighbor information. Sensors then determine the existence of coverage holes by examining their Voronoi polygons. If any hole exists, sensors calculate where to move to eliminate or reduce the size of the coverage hole. Three algorithms are proposed to calculate the target locations: VEC *pushes* sensors away from a densely covered area; VOR *pulls* sensors to the sparsely covered area; and Minimax moves

sensors to the center of their Voronoi polygon. Termination conditions are defined for each algorithm.

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 exerts forces, denoted as \vec{F}_b , to push sensors too close to the boundary inward. \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 “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 or 3/4 point 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.

Notations:

$\mathcal{N}_i, G_i, \vec{F}_{ij}, \vec{F}_b, d_{ave}$: defined before
 c_i : whether G_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** $c_i = \text{false}$ **then**
call $VEC()$ /* call VOR and Minimax
in other protocols */
 - (2.3) Done when satisfying stop criteria
 - (3) Upon receiving a *hello* message from sensor s_j :
 - (3.1) Update \mathcal{N}_i and G_i
 - (3.2) **if** G_i is newly covered **then**
set $c_i = \text{true}$
Broadcast *OK* /* only for VEC */
- /* The following is only for VEC ,
and will be replaced in VOR and Minimax */
- (4) Upon receiving an *OK* message from sensor s_j :
 - (4.1) set $c_j = \text{true}$
 - (5) $VEC()$
 - (5.1) $\vec{v}_i = \vec{0}$
 - (5.2) **for each** s_j in \mathcal{N}_i
 - if** $(c_j \neq \text{true}) \wedge (d_{ave} > d(s_i, s_j))$ **then**
 $|\vec{F}_{ij}| = (d_{ave} - d(s_i, s_j))/2$; $\vec{v}_i = \vec{v}_i + \vec{F}_{ij}$
 - if** $(c_j = \text{true}) \wedge (d_{ave} > d(s_i, s_j))$ **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**

Figure 3. The VEC protocol at sensor s_i

Figure 2 shows an operational example of VEC. Round 0 is the initial random deployment of 35 sensors in a 50m by 50m flat space, with the sensing range of 6 meters and communication range of 20 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 Figure 3.

B. The VORonoi-based Algorithm (VOR)

Contrary to the VEC algorithm, VOR is a *pull* algorithm which pulls sensors to cover their local maximum coverage holes. In VOR, if a sensor detects the existence of coverage holes, it will move toward its farthest

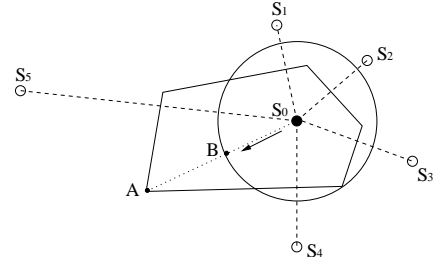


Figure 4. VOR

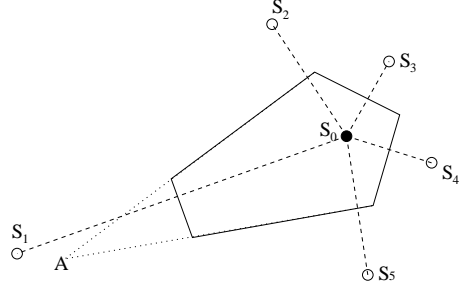


Figure 5. Inaccurate Voronoi polygon

Voronoi vertex (denoted as V_{far}), and stop when the farthest Voronoi vertex can be covered. Same as VEC, in VOR, a sensor needs only to check its own Voronoi polygon. Figure 4 illustrates VOR. Point A is the farthest Voronoi vertex of s_0 , and $d(A, s_0)$ is longer than the sensing range. To heal the hole, s_0 moves along line s_0A 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 minus the sensing range to avoid the situation shown in Figure 5, in which s_0 is not aware of the existence of s_1 because of communication limitations. When s_0 does not know s_1 , it will calculate its local Voronoi polygon as the dotted one and view the area around A as a coverage hole. If s_0 moves toward point A and stops at a distance $d(A, B)$ (sensing range), apparently s_0 has moved more than needed and it tries to cover the area which is already covered by s_1 . It is quite possible it has to move back after it gets to know s_1 . Therefore, we set the maximum moving distance such that a sensor moves towards the coverage hole step-by-step. After s_0 moves certain distance, and it gets closer to s_1 , it can communicate with s_1 and calculate the correct Voronoi polygon. Then the risk of moving oscillation can be greatly reduced.

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. Before a sensor moves, it first checks whether its moving direction is opposite to that in the previous round. If yes, it stops for one round.

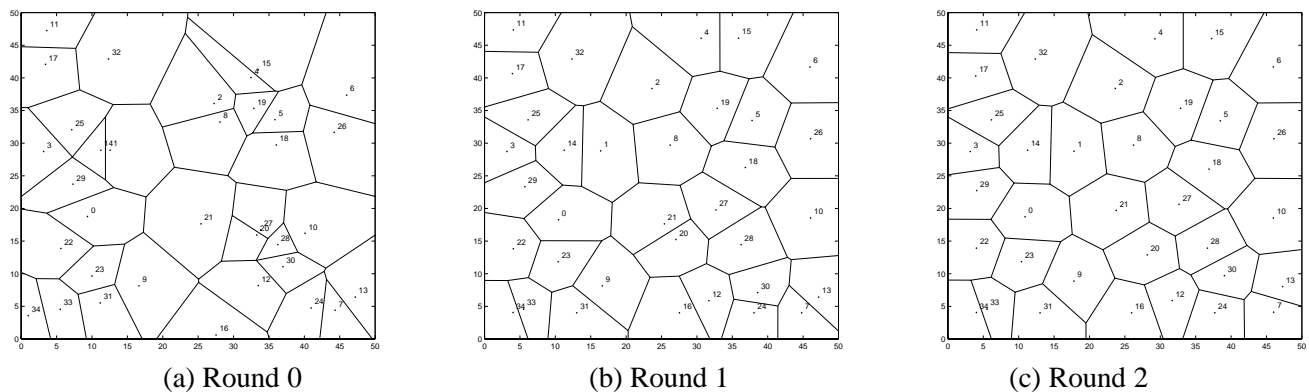


Figure 2. Snapshot of the execution of VEC

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**
-

We run VOR on the same initial deployment as shown in Figure 2(a). 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 situations in which a vertex that 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 Minimax point is the center of the smallest enclosing circle of the Voronoi vertices and can be calculated by the algorithms described in [19], [27], [29]. In the deployment protocol using Minimax, we also specify the maximum moving distance, and do oscillation control as in VOR.

We run Minimax on the same initial deployment as shown in Figure 2(a). 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 coverage cannot be increased. The formal proof is shown in APPENDIX.

In some applications, the coverage requirement may be met without achieving maximum coverage. In this case, it may be prudent 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

1) *Dealing With Message Loss:* Hello messages may be lost due to collisions. Consequently, sensors may fail to know the existence of some Voronoi neighbors and mistakenly determine coverage holes. To address this problem, we associate each item in a sensor's neighbor list

with a number which indicates the freshness of this item. That is, for how many rounds this neighbor has not been heard. When constructing the Voronoi polygon, sensors only consider the sensors in its neighbor list with certain freshness. For example, only sensors that have been heard within the last two rounds can be considered when constructing the Voronoi polygon. Supposing the probability of message loss is 5%, the probability that a message is lost two consecutive times is 0.25%. Therefore, if a sensor does not hear a HELLO message from a neighbor for two consecutive cycles, it can assume that the neighbor has moved and be correct with a 99.75% chance.

This solution introduces a new problem. If a sensor actually moves to a new place, its previous neighbors cannot hear it. If these old neighbors still consider this sensor in their formation of the Voronoi polygons until the freshness threshold is violated, it will prolong the deployment process. To address this problem, we propose that a sensor broadcasts its new location before it moves so that its neighbors can react promptly if a new hole is generated by its leaving.

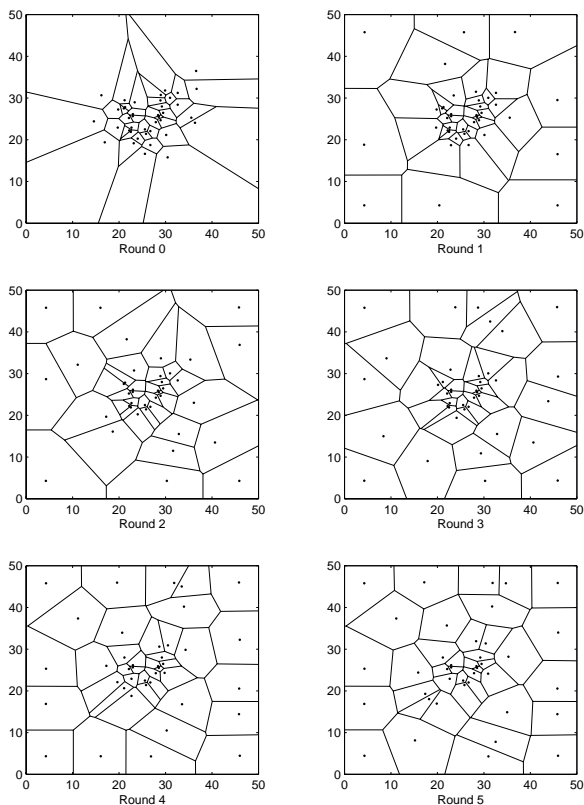


Figure 6. Working procedure (VOR)

2) *Dealing with Position Clustering:* In some cases, the initial deployment of sensors may form clusters, as shown in Figure 6, 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, as is shown in Figure 6, in which 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. 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. DEPLOYMENT PROTOCOLS WITH VIRTUAL MOVEMENT

The basic 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. One such approach 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, virtually move there, and exchange these new virtual locations with the sensors which would be their neighbors as if they had actually moved. The real movement only happens at the last round after final destinations are determined.

We did not deploy this alternative method for two reasons. First, this approach 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 $2rn^2$ (here r is the number of rounds needed and n is the number of sensors in the network). Using actual mobility as in the basic protocols, a much lower message complexity, $2rn$, is sufficient.

To get balance between movement and message complexity, we propose to let sensors do virtual movement

when the communication cost to reach the logical Voronoi neighbors is reasonable, and do physical movement otherwise. The challenge is to determine if a sensor can reach its logical neighbors with reasonable communication cost. We propose the following heuristics.

First, if a sensor's distance to its farthest Voronoi vertex is shorter than half of the communication range, it must know all its Voronoi neighbors. In this case, one hop broadcast (same in the basic protocols) is enough to exchange the location information with its logical neighbors and physical movement is not necessary. Otherwise it is possible that some Voronoi neighbors are out of the communication range. To get the locations of these Voronoi neighbors, sensors request their neighbors within the communication range to broadcast their neighbor lists, thus obtaining the logical positions of sensors located within two broadcast hops. When the distances between the physical locations of sensors and their farthest Voronoi Vertices are larger than two times the maximum moving distance, sensors should move physically.

In realization, we divide the discovery phase into two sub-phases. In the first sub-phase, sensors broadcast hello messages; in the second sub-phase, sensors broadcast the locations of known neighbors. In one round, if a sensor's distance to its farthest Voronoi vertex is larger than half of the communication range, it will calculate the target location as in the basic schemes, and do logical movement. In the next round, it will set a flag in the hello messages, indicating that it wants its neighbors to broadcast their neighbor list. Any sensor that receives a hello message with such a flag will broadcast its neighbor list in the second sub-phase of the discovery phase. In this way, the message complexity is at most two times the basic scheme in one round. The flag will not be reset until the sensor moves physically. Sensors move physically under two conditions: One is that its physical position is two times the maximum moving distance to its farthest Voronoi vertex, as discussed above. The second condition is that a sensor's logical position has not changed for several rounds. Then the sensor can determine that it has obtained its final location and it can move. The formal description of the protocol with the virtual movement is shown in Figure 7.

V. PERFORMANCE EVALUATIONS

A. Objectives, Metrics, and Methodology

We implement our deployment protocols in the ns-2 (version 2.1b9a), a standard network simulator. Our objectives in conducting this evaluation study are three-fold: first, testing the effectiveness of our protocols in providing high coverage; second, by comparing VEC, VOR and

Notations:

- d_{max}, c_i : defined before
 - d_c : communication range
 - NL_i : the neighbor list of s_i
 - w_i : whether s_i wants the neighbor list of its neighbors
 - l_i : whether s_i needs to broadcast its neighbor list
 - loc_i, loc'_i : the physical and logical position of s_i
 - p_i : loc'_i has not changed for p_i rounds.
 - P : s_i should move if $p_i \leq P$
- (1) *Upon entering Discovery phase-I:*
 - (1.1) set *timer* to be $discovery_interval/2$
enter *Discovery phase-II* upon timeout
 - (1.2) broadcast $hello(w_i)$ after a random time slot
 - (2) *Upon entering Discovery phase-II:*
 - (2.1) set *timer* to be $discovery_interval/2$
enter *Moving phase* upon timeout
 - (2.2) **if** $l_i = true$ **then**
broadcast NL_i after a random time slot
 $l_i = false$
 - (3) *Upon entering Moving phase:*
 - (3.1) set *timer* to be $moving_interval$,
enter *Discovery phase-I* upon timeout
 - (3.2) **if** $c_i = false$ **then**
calculate loc'_i by VEC or VOR or Minimax
do oscillation control
do movement adjustment
 $p_i = 1$ if logically moves
if $d(loc_i, V_{far}) \geq 2 * d_{max}$ **then**
move to loc'_i
 $w_i = false$;
else if $d(loc'_i, V_{far}) \geq d_c/2$ **then**
 $w_i = true$;
 - (3.3) **else if** $p_i \leq 1$ **then**
 $p_i = p_i + 1$
 - (3.4) **else if** $p_i \leq P$
move to loc'_i
 $w_i = false$;
 - (4) *Upon receiving a $hello(w_j)$ message from sensor s_j :*
 - (4.2) **if** $w_j = true$ **then**
 $l_i = true$
 - (4.1) Update \mathcal{N}_i and G_i
 - (4.2) **if** G_i is newly covered **then**
set $c_i = true$
 - (5) *Upon receiving a NL_j message from sensor s_j :*
 - (5.1) Update \mathcal{N}_i and G_i

Figure 7. Virtual movement protocols at s_i

Minimax, and comparing the basic protocols and the virtual movement protocols, giving some insight on choosing protocols in different situations; finally, studying the effectiveness of controlling the tradeoff among various metrics by adjusting parameters.

We analyze the performance of our protocols from two aspects: *deployment quality* and *energy consumption*. *Deployment quality* is measured by the sensor coverage and the time (number of rounds) 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. Energy consumption includes two parts, mechanical movement and communication. Message complexity is used to measure the energy consumed in communication. As for movement, the energy consumed in moving a sensor n meters consists of two parts: starting/braking energy and moving energy. Therefore, we use moving distance and the number of movement as the metrics.

We run simulations under different sensor density, which determines the sensor coverage that can be reached and the difficulty to reach it. In a $100m * 100m$ target field, we distribute four different number of sensors, ranging from 120 to 180, in increments of 20 sensors. The initial deployment follows the random distribution. Most simulation results are under the termination condition that ϵ is equal to 1% divided by the number of sensors. To evaluate each metric under different parameter settings, we run 10 experiments based on different initial distribution and calculate the average results.

We choose 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. Based on the information from [1], 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)[2].

B. Simulation Results

1) *Coverage*: Figure 8 shows the coverage obtained when the coverage increase threshold ϵ is equal to 1% divided by the number of sensors. From the figure, we can see that the coverage is greatly increased by all three algorithms compared to the initial random distribution. For example, when 140 sensors are deployed, Minimax and VOR can increase the coverage to be more than 98% from

77.7%. In contrast, to obtain the same coverage under random deployment, on average, 340 sensors are required.

Also, by analyzing the trace of the basic protocols, we find the coverage increases very quickly during the first several rounds. For example, in most of the cases, the coverage can be increased to over 85% after the first round when 120 sensors are deployed and over 90% in higher sensor densities. In the virtual movement protocols, actual coverage increase happens after the real movement.

Among VEC, VOR, and Minimax, VEC performs the worst. The primary reason is that VEC is sensitive to the initial deployment. Consider an extreme situation in which sensors are located in the same line with equal spacing. 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 similar relative positions initially, VEC does not perform well. In addition, VEC neither considers coverage holes nor utilizes any geometric information from the Voronoi polygons when choosing the target locations. It tries to reach relatively balanced positions among the sensors, despite the difficulty of obtaining an exact global even distribution from only local information.

VOR and Minimax achieve quite similar coverage. They both move to heal the holes directly. VOR is more greedy and may move more than needed, thus generating new coverage holes. But finally, VOR will move sensors back to the correct positions if coverage can be increased by doing so.

Between virtual movement protocols and basic protocols, virtual movement protocols achieve almost the same coverage as the basic protocols as expected. We can conclude that using virtual movement will not affect the achieved coverage.

2) *Energy Consumption*: Figure 9 and Figure 10 show the moving distance and the number of movements respectively. Figure 11 shows the message complexity. Here message complexity is defined as the number of messages exchanged when the protocol terminates. To evaluate the energy consumption, we normalize the moving distance and the number of movements into message complexity. That is, with the same amount of energy consumed in movement, how many messages can be transmitted. Calculated from Robomote [26], approximately, to move a sensor one meter consumes a similar amount of energy as transmitting 300 messages. The energy consumption in starting/braking is varied in different systems. Figure 12 shows the unified energy consumption when the starting/braking to one meter moving energy consumption ratio is one. (We also have plotted the case when the ratio is four. The results are similar to Figure 12, so we do not

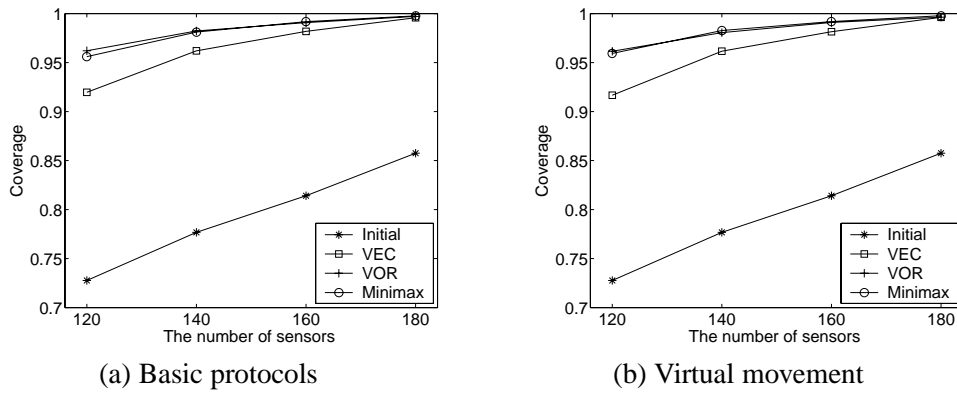


Figure 8. Coverage

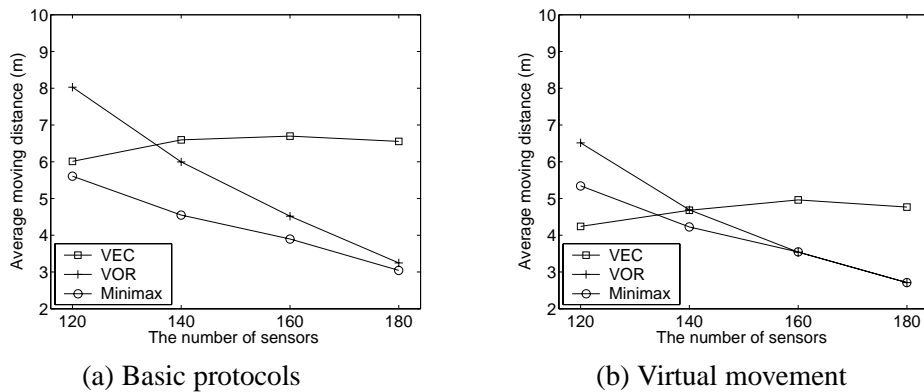


Figure 9. Moving distance

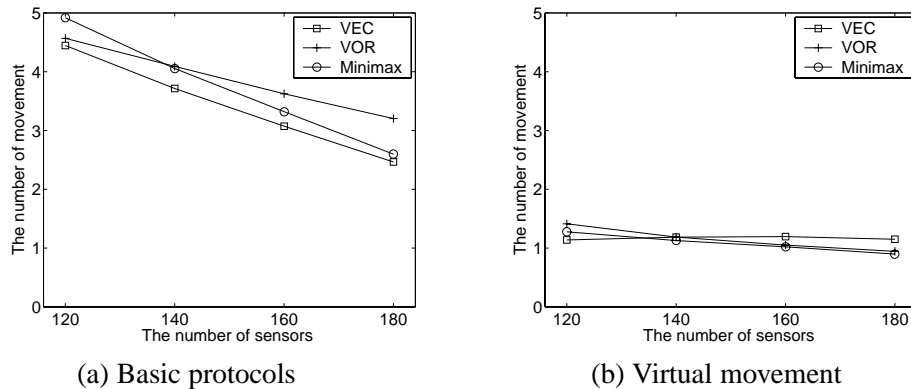


Figure 10. The number of movements

show it here.) From the figure, we can conclude that virtual movement protocols are much more energy-efficient than the basic protocols. The improvement is larger when the starting energy is high. Among the three algorithms to calculate the target locations, Minimax is always the most energy-efficient, except when the sensor density is quite low. At low sensor density, VEC consumes the least energy. VEC pushes sensors into relatively regular positions and does not perform the fine adjustment of their locations to reach high coverage. Therefore, VEC consumes the least energy, and reaches the lowest coverage among

these three algorithms under low density.

Between VOR and Minimax, Minimax moves less in most cases. Minimax is proposed to address the aggressive feature of VOR and the simulation results verify its effectiveness. Because Minimax is sensitive to the construction of Voronoi polygon, it is sensitive to message loss. In our previous paper [10], we showed that if message loss is not accounted for, Minimax has the largest moving distance. This is because when message loss occurs, the calculated Voronoi polygon is not correct.

From Figure 9 we can observe an interesting phenomena of VEC: the moving distance is similar under differ-

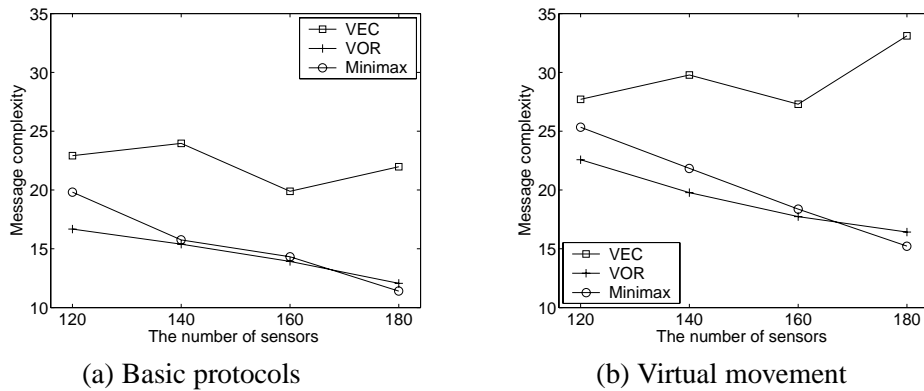


Figure 11. Message Complexity

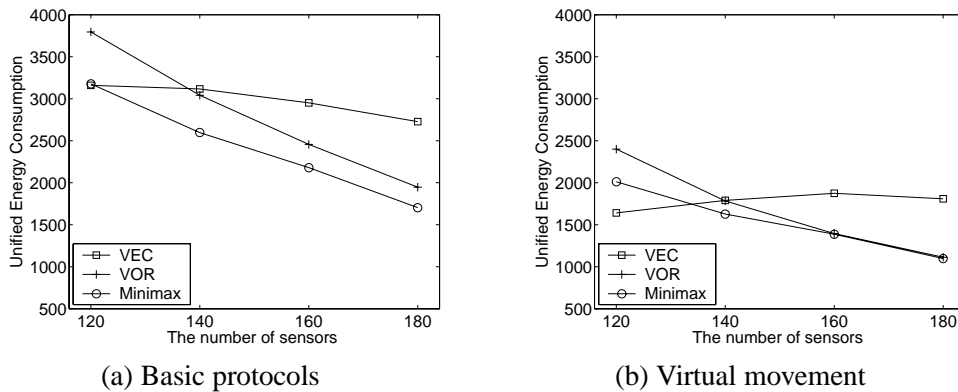


Figure 12. Unified Energy Consumption (energy consumed in starting/braking is equal to moving one meter)

ent 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, VEC is not sensitive to sensor density. In contrast, Minimax and VOR relocate sensors by measuring the coverage holes, which are larger under lower density and smaller under high density. Therefore, the moving distance in VOR and Minimax is decreased with a higher sensor density.

Figure 10 shows the number of movements using basic protocols and virtual movement protocols, respectively. We can see that the number of movements is greatly reduced by using virtual movement. In particular, sensors move about once when the number of nodes is more than 140, and less than 1.5 times when the number of nodes is 120. This shows the effectiveness of the heuristic to determine when to move physically. In the basic protocols, sensors move several times on average. They move less with a higher node density and fewer coverage holes.

As described in Section II-B, path planning is required to overcome obstacles. We expect a greater negative im-

pact on moving distance when obstacles must be overcome on the protocols that require the larger number of movements.

Figure 11 shows the message complexity. Message complexity is primarily determined by the number of rounds to finish the deployment process and the number of messages in each round. Within one round, VEC transmits more messages than VOR and Minimax since sensors need to send one message to notify neighbors that their Voronoi polygons are well covered. In addition, VEC needs more rounds to terminate (explained in the next section). Therefore, VEC has the highest message complexity. Between Minimax and VOR, Minimax needs more rounds to terminate and has higher message complexity.

3) *Convergence Time*: In this section, we evaluate the convergence time of our protocols. We set ϵ to be 0. Figure 13 shows the coverage in each round when the number of sensors is 140. From the figure, we can see that the coverage increases very quickly during the first several rounds. Here for each algorithm, we run 50 experiments. After 10 rounds, the algorithms achieve at least 98% of the best coverage they can reach, in all these experiments and the algorithms achieve at least 99% of the best coverage, in about 99.33% of these experiments. From the experiments, we can conclude that our algorithm can quickly

converge.

Our algorithm resembles the steepest descent algorithm [16], in that both try to move along the direction that is locally optimal. Unfortunately, the convergence theory of steepest descent is not satisfactory from a theoretical point of view. It is shown in [16] that it converges if the objective function satisfies certain conditions and proper step lengths are taken. However, no result on the convergence rate of steepest descent for general objective functions can be found in the literature. For most applications, steepest descent is the best choice if no global information of the objective function is available.

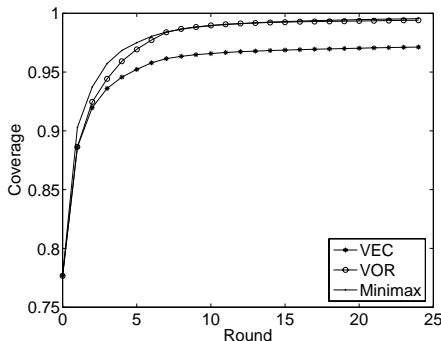


Figure 13. Convergence time

4) *Termination*: Figure 14 shows deployment time under different sensor density. As expected, virtual movement protocols require more time to terminate. In virtual movement, each sensor waits several rounds before real movement.

In general, the deployment procedure can be roughly divided into two parts: One is overall re-distribution, which moves a group of sensors from a dense area to a sparse area to achieve a relatively even distribution of sensors. Another is the minor adjustment of positions in the local area to achieve better coverage. When the sensor density is not high, time consumed in the overall re-distribution is the major factor of deployment. When the sensor density is very high, no large-scale movement is needed and the position adjustment is the major factor. VOR is good at the overall re-distribution because of its aggressive feature. It terminates the quickest in low or medium sensor density. Minimax calculates the target locations to heal coverage holes most accurately, and it finishes the quickest in very high density. VEC pushes sensors away from dense area by virtual force. The sensors in a sparse area may not move for a long time since no sensor is present to push them. These sensors only move after the sensors from a dense area are propagated into their area. This propagation process may take a long time.

5) *Impact of Coverage Increase Threshold ϵ* : In this section, we study the effectiveness of controlling the

tradeoff between coverage and deployment time by adjusting ϵ , the coverage increase threshold. TABLE I shows the termination round, coverage reached and other metrics for different ϵ for the virtual movement protocols. We can see that, with a smaller ϵ , a higher coverage can be reached, while the deployment cost and the deployment time is also increased. By properly setting this threshold, we can save time and energy by trading off a small amount of coverage. For example, in Minimax, when ϵ is increased from 0.5% to 1%, the deployment time can be shortened by 7 rounds (32%), while the ultimate coverage achieved is only reduced by 0.4%.

Among VEC, VOR, and Minimax, the termination time is quite different when ϵ is small, and similar when ϵ is larger. For example, when ϵ is equal to 2%, the three algorithms terminate in a similar time. As for other metrics, Minimax performs the best. Therefore, Minimax is the best choice if ϵ can be set to be a relatively large number.

VI. CONCLUSION AND DISCUSSIONS

This paper addressed the problem of moving sensors in a target field to get high coverage. Based on Voronoi diagrams, we designed two sets of distributed protocols to iteratively move mobile sensors from densely deployed areas to sparsely deployed areas. Simulation results verified the effectiveness of our protocols and provided a baseline for performance under ideal conditions.

The virtual movement protocols can significantly reduce mechanical movement with a cost of less than two times an increase in the message complexity over the basic protocols. In each set of the protocols, three algorithms to calculate the target location were proposed: VEC, VOR and Minimax. Minimax is the best choice in most cases. VEC moves least at a low sensor density and can be deployed when the coverage requirement is not high. VOR terminates the earliest when the sensor density is not very high, and it can be deployed when both the deployment time requirement and coverage requirement are strict.

Below, we discuss some open issues.

A. Distributed Scheme versus Centralized Scheme

To address the problem of mobile sensor deployment, we propose that sensors calculate their target locations in a distributed fashion. We did not deploy a *centralized* approach for the following reasons. First, a central server architecture may not be feasible in some deployments. Further, the centralized approach suffers from the problem of a single point of failure.

Although a *centralized* approach is not feasible for many scenarios, it is interesting to study it and compare

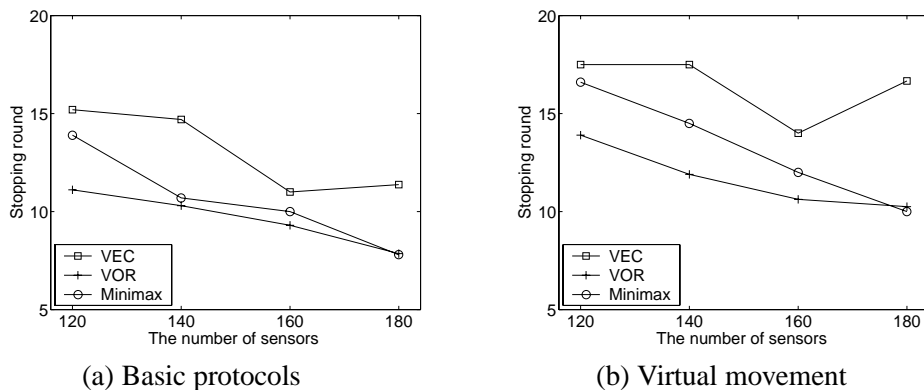


Figure 14. Termination

| $\epsilon * n$ | VEC | | | | | VOR | | | | | Minimax | | | | |
|----------------|-------|-------|------|------|------|-------|-------|------|------|------|---------|-------|------|------|------|
| | R | C(%) | D(m) | M | E | R | C(%) | D(m) | M | E | R | C(%) | D(m) | M | E |
| 0.5% | 20.60 | 96.41 | 4.72 | 1.21 | 1.10 | 14.70 | 98.52 | 4.78 | 1.24 | 1.06 | 21.20 | 98.65 | 4.30 | 1.17 | 1.05 |
| 1% | 17.50 | 96.16 | 4.68 | 1.19 | 1.10 | 11.90 | 98.06 | 4.69 | 1.19 | 1.05 | 14.50 | 98.27 | 4.22 | 1.13 | 1.04 |
| 2% | 11.40 | 95.79 | 4.63 | 1.17 | 1.09 | 10.20 | 97.46 | 4.61 | 1.16 | 1.04 | 10.70 | 97.87 | 4.15 | 1.10 | 1.03 |

TABLE I

IMPACT OF ϵ ($n = 140$) (**R** is the round number when all sensors stop. **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. These values are obtained in the stopping round **R**)

it with our distributed algorithms. We consider a centralized approach in an ideal case, in which the optimal positions to place sensors is decided a priori. For n sensors, there are n regular positions. It does not matter which sensor is placed in which position. There exists a central server, which can collect the locations of sensors and direct them to move, such that the average moving distance is minimized. Basically, this is a bi-party matching problem and the classic Hungarian method [23] can be used to calculate how to allocate sensors from their initial positions to the target locations such that the moving distance is minimized. We have compared the average moving distance of our distributed protocols with this ideal case. When the node density is lower than about 110 sensors per $100m * 100m$, the centralized scheme outperforms our scheme; otherwise, our distributed algorithms are better. (Here, we choose Minimax under virtual movement for comparison). This is because in our scheme sensors only move when there are coverage holes, while in the centralized scheme sensors always move to an optimal point even if it does not improve coverage. In our scheme, when the node density is high, sensors need only move a short distance to reach high coverage. Under low density when most sensors are required to move, the centralized scheme results in a lower average moving distance. In terms of coverage, the centralized approach can always guarantee the optimal coverage since the optimal positions are decided a priori.

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. 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 protocols 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 polygons.

If the sensing area is an irregular shape, instead of a disk, sensors can still check their Voronoi polygons 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. Sensitivity to Communication Range

In our previous paper [10], we evaluated the impact of communication range on the basic protocols and found that when communication is more than two times of the sensing range, the performance is similar to the results

presented here (6m sensing range and 20m communication range). This requirement is reasonable and most hardware can satisfy it. In situations in which this requirement cannot be satisfied, we can increase the broadcast hop limit. In the virtual movement protocols, the broadcast hop can be increased accordingly if the communication range is short and the performance will not be affected.

APPENDIX

We denote the following terms for the proof. The location of sensor s_i in the r^{th} round is denoted as $[x_i^{(r)}, y_i^{(r)}]$. The Voronoi polygon of s_i , G_i , in the r^{th} round is denoted as $G_i^{(r)}$. G_i changes in different rounds if s_i or its neighbors move. Area of the covered part of $G_i^{(r)}$ in the r^{th} round is denoted as $A_i^{(r)}$ and that in the $(r+1)^{th}$ round is denoted as $\hat{A}_i^{(r)}$. $G_i^{(r)}$ is a fixed polygon in the target field, but the covered portion of $G_i^{(r)}$ may be changed in different rounds since sensors may move and they cover different areas if they move. Therefore, $\hat{A}_i^{(r)}$ is not equal to $A_i^{(r)}$. Also, $\hat{A}_i^{(r)}$ is not equal to $A_i^{(r+1)}$. They refer to the covered area of different polygons. The area of the covered portion of $G_i^{(r)}$ by a sensor located at $[x, y]$ is denoted as $A_i^{(r)}([x, y])$. The area of the covered portion in the whole target field in the r^{th} round is $A_{total}^{(r)}$.

$$\begin{aligned} \text{Lemma 1: (a)} & A_{total}^{(r)} = \sum_{i=1}^n A_i^{(r)}; \\ \text{(b)} & A_{total}^{(r+1)} = \sum_{i=1}^n \hat{A}_i^{(r)}. \end{aligned}$$

Proof: Voronoi diagram is a partition of the target field, so (a) is obvious. With the same reason, (b) is also correct. The summation of the covered area of partitions is the whole covered area in the target field, whatever a partitioning method is used. Therefore, the summation of the covered area of the Voronoi polygons in the previous round is also the whole covered area in the current round. ■

$$\text{Lemma 2: } A_i^{(r)} = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}])$$

Proof: This is the direct result of the attribute of Voronoi diagram. Every point within $G_i^{(r)}$ is closer to $[x_i^{(r)}, y_i^{(r)}]$ than to any other sensor. Any point not covered by s_i is also not be covered by any other sensor. ■

Theorem 1: $A_{total}^{(r+1)} > A_{total}^{(r)}$ before all sensors stop moving.

Proof: At the $(r+1)^{th}$ round, there may be areas in $G_i^{(r)}$ which is not covered by s_i , but is covered by other sensors, because the current Voronoi polygon of s_i

is $G_i^{(r+1)}$, but not $G_i^{(r)}$. Therefore,

$$\hat{A}_i^{(r)} \geq A_i^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) \quad (1)$$

By enforcing the *movement adjustment* heuristics (described in section III-A), our algorithms guarantee that, if s_i moves,

$$A_i^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) > A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]). \quad (2)$$

Certainly, if s_i does not move,

$$A_i^{(r)}([x_i^{(r+1)}, y_i^{(r+1)}]) = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]), \quad (3)$$

because $[x_i^{(r+1)}, y_i^{(r+1)}] = [x_i^{(r)}, y_i^{(r)}]$.

From (1),(2),(3)

$$\sum_{i=1}^n \hat{A}_i^{(r)} > \sum_{i=1}^n A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]), \quad (4)$$

if some sensor moves in the r^{th} round.

From Lemma 2

$$A_i^{(r)} = A_i^{(r)}([x_i^{(r)}, y_i^{(r)}]) \quad (5)$$

From(4),(5)

$$\sum_{i=1}^n \hat{A}_i^{(r)} > \sum_{i=1}^n A_i^{(r)}, \quad (6)$$

if some sensor moves in the r^{th} round.

By Lemma 1,

$$\begin{cases} A_{total}^{(r)} = \sum_{i=1}^n A_i^{(r)} \\ A_{total}^{(r+1)} = \sum_{i=1}^n \hat{A}_i^{(r)} \end{cases} \quad (7)$$

From (7),(6)

$$A_{total}^{(r+1)} > A_{total}^{(r)}, \quad (8)$$

if some sensor moves in the r^{th} round. ■

Corollary 1: Our distributed algorithms are convergent, and thereby terminate naturally.

Proof: Following from Theorem 1 and the fact that $A_{total}^{(r)}$ is upper bounded by the total area of the target field, our distributed algorithms converge, and terminate naturally. All sensors stop moving when no coverage increase can happen. ■

Acknowledgments

This work was supported in part by the National Science Foundation CNS-0519460 and a grant from MARCO/DARPA Gigascale Systems Research Center. The authors also want to thank Dr. Sergio Palazzo and the anonymous reviewers for their helpful comments on an early version of this paper. Preliminary results of this paper have been presented in part in IEEE INFOCOM 2004 in Hong Kong [10].

REFERENCES

- [1] “Berkeley Sensor and Actuator Center,” <http://www-bsac.eecs.berkeley.edu>.
- [2] “Wireless Sensing Networks,” <http://wins.rsc.rockwell.com>.
- [3] “US Naval Observatory (USNO) GPS Operations,” <http://tycho.usno.navy.mil/gps.html>, April 2001.
- [4] F. Aurenhammer, “Voronoi Diagrams — A Survey of a Fundamental Geometric Data Structure,” *ACM Computing Surveys*, vol. 23, pp. 345–405, 1991.
- [5] T. Clouqueur, V. Phipatanasuphorn, P. Ramanathan and K. K. Saluja, “Sensor Deployment Strategy for Target Detection,” *First ACM International Workshop on Wireless Sensor Networks and Applications*, 2002.
- [6] D. Koditschek, “Planning and Control via Potential Functions,” *Robotics Review I*, pp. 349–367, 1989.
- [7] D. Niculescu and B. Nath, “Ad Hoc Positioning Systems (APS) Using AoA,” *IEEE INFOCOM*, 2003.
- [8] F. Zhao and L. Guibas, *Wireless Sensor Networks*, Morgan Kaufmann, 2004.
- [9] S. Fortune, D. Du and F. Hwang, “Voronoi Diagrams and Delaunay Triangulations,” *Euclidean Geometry and Computers*, 1992.
- [10] Guiling Wang, Guohong Cao and Tom La Porta, “Movement-Assisted Sensor Deployment,” *IEEE INFOCOM*, March 2004.
- [11] W. R. Heinzelman, J. Kulik and H. Balakrishnan, “Adaptive Protocols for Information Dissemination in Wireless Sensor Network,” *ACM MobiCom*, 1999.
- [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] 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.
- [14] L. Hu and D. Evans, “Localization for Mobile Sensor Networks,” *ACM MobiCom*, 2004.
- [15] C. Intanagonwivat, R. Govindan and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication,” *ACM MobiCom*, 2000.
- [16] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
- [17] J. Lengyel, M. Reichert, B. Donald, and D. Greenberg, “Real-time Robot Motion Planning using Rasterizing Computer Graphics Hardware,” *Proc. SIGGRAPH*, 1990.
- [18] Q. Li, M. De Rosa, and D. Rus, “Distributed Algorithms for Guiding Navigation Across a Sensor Network,” *ACM MobiCom*, 2003.
- [19] N. Megiddo, “Linear-time Algorithms for Linear Programming in R^3 and Related Problems,” *SIAM J. Computing* 12, pp. 759–776, 1983.
- [20] S. Meguerdichian, F. Koushanfar, G. Qu and M. Potkonjak, “Exposure In Wireless Ad-Hoc Sensor Networks,” *ACM MobiCom*, 2001.
- [21] S. Meguerdichian, F. Koushanfar, M. Potkonjak and M. B. Srivastava, “Coverage Problems in Wireless Ad-hoc Sensor Network,” *IEEE INFOCOM*, 2001.
- [22] N. Patwari and A. Hero III, “Using Proximity and Quantized RSS for Sensor Location in Wireless Location in Wireless Networks,” *Workshop on Wireless Sensor Networks and Applications*, 2003.
- [23] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, 1998.
- [24] G. J. Pottie and W. J. Kaiser, “Wireless Integrated Network Sensors,” *Communications of the ACM*, May 2000.
- [25] A. Savvides, C. Han and M. B. Srivastava, “Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors,” *ACM MobiCom*, 2001.
- [26] G. T. Sibley, M. H. Rahimi and G. S. Sukhatme, “Robomote: A Tiny Mobile Robot Platform for Large-Scale Sensor Networks,” *2002 IEEE International Conference on Robotics and Automation*, 2002.
- [27] S. Skyum, “A Simple Algorithm for Computing the Smallest Enclosing Circle,” *Information Processing Letters*, vol. 37, pp. 121–125, 1991.
- [28] K. Sohrabi, J. Gao, V. Ailawadhi and G. J. Pottie, “Protocols for Self-Organization of A Wireless Sensor Network,” *IEEE Personal Communication*, vol. 7, no. 5, pp. 16–27, October 2000.
- [29] E. Welzl, “Smallest Enclosing Disks (Balls and Ellipsoids),” *New Results and New Trends in Computer Science, volume 555 of LNCS*, pp. 359–370, 1991.
- [30] Y. Zou and K. Chakrabarty, “Sensor Deployment and Target Localization Based on Virtual Forces,” *IEEE INFOCOM*, 2003.

Guiling Wang received her BS degree from Nankai University, China. She is currently pursuing the PhD degree in Computer Science and Engineering at the Pennsylvania State University. Her research interests include distributed systems, wireless networks and mobile computing, with a focus on wireless sensor networks. She is a student member of the IEEE.

Guohong Cao received his BS degree from Xian Jiaotong University, Xian, China. He received the MS degree and PhD degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently an Associate Professor. His research interests are wireless networks and mobile computing. He has published one hundred papers in the areas of sensor networks, cache management, data dissemination, resource management, wireless network security, and distributed fault-tolerant computing. He is an editor of the IEEE Transactions on Mobile Computing and IEEE Transactions on Wireless Communications, a co-guest editor of special issue on heterogeneous wireless networks in ACM/Kluwer Mobile Networking and Applications, and has served on the program committee of many conferences. He was a recipient of the Presidential Fellowship at the Ohio State University in 1999, and a recipient of the NSF CAREER award in 2001.

Thomas F. La Porta received his Ph.D. degree in Electrical Engineering from Columbia University, New York, NY. He joined the Computer Science and Engineering Department at Penn State in 2002 as a Full Professor. He is the Director of the Networking and Security Research Center at Penn State. Prior to joining Penn State, Dr.

La Porta was with Bell Laboratories since 1986. There he was the Director of the Mobile Networking Research Department where he led various projects in wireless and mobile networking. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award in 1996.

Dr. La Porta was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing. He also served as Editor-in-Chief of IEEE Personal Communications Magazine for three years. He has published over 50 technical papers and holds 28 patents. He was an adjunct member of faculty at Columbia University for 7 years where he taught courses on mobile networking and protocol design.