Non-Uniform Sensor Deployment in Mobile Wireless Sensor Networks

Mihaela Cardei, Yinying Yang, and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
Email: {mihaela@cse., yyang4@, jie@cse.}fau.edu

Abstract*—In wireless sensor networks (WSNs), good sensor deployment is vital to the coverage of the monitored area and to the network lifetime. To improve the initial deployment, one possible method is to use mobile sensors, thus allowing sensors to relocate. In order to prolong network lifetime and achieve balanced energy consumption, one approach is to place sensors in different densities which vary with the distance to the sink. Since sensors located closer to the sink are involved in more data forwarding, sensors in this region should have a higher density. Movement-assisted sensor deployment involves moving sensors with the purpose of meeting the density requirements according to their distance to the sink. The additional requirement of coverage is also discussed. In this paper, we address the problem of Movement-assisted Sensor Positioning (MSP) to increase network lifetime with the objective to achieve the theoretical sensor densities while minimizing sensor movement. We propose three solutions to the MSP problem: an Integer-Programming formulation, a localized matching method, and a distributed corona-radius scanning algorithm. Simulation results are presented to verify the effectiveness of the proposed solutions.

I. Introduction

A wireless sensor network (WSN) consists of a large number of sensor nodes that are densely deployed either inside the phenomenon or very close to it [1]. Sensor nodes measure various parameters of the environment and transmit data collected to one or more sinks using hop-by-hop communication. Once a sink receives sensed data, it processes and forwards it to the users. In mobile sensor networks, sensors can self-propel, can move using wheels [4], springs [2], or they can be attached to transporters such as robots [4] and vehicles [6].

A large number of sensors can be distributed in mass by scattering them from airplanes, rockets, or missiles [1]. The initial deployment is hard to control using such deployment mechanisms. However, a good deployment is vital in order to improve coverage, achieve load balance, and prolong the network lifetime. In general, there are two methods used to improve the initial deployment: by deploying additional sensors [5] or by movement-assisted sensor positioning mechanisms [2], [11], [13], [14].

In a WSN, sensors closer to the sink tend to consume more energy than those farther away from the sink. This is mainly because, besides transmitting their own packets, they forward packets on behalf of other sensors that are located farther away. If sensors in the network are uniformly distributed, then the sensors closer to the sink will deplete their energy resources

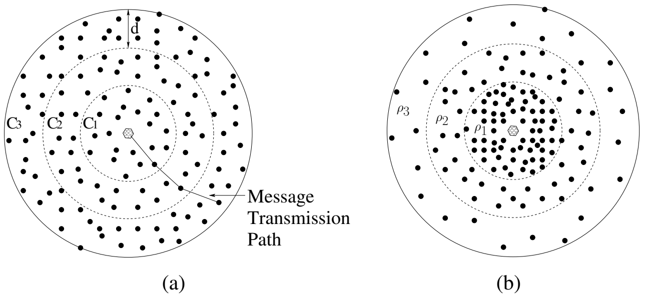


Fig. 1. WSN model using coronas concentric to the sink. (a) Uniform distribution, (b) Non-uniform distribution.

first [8], resulting in holes in the WSN. This uneven energy consumption will reduce network lifetime.

In [8], Olariu *et al.* consider a uniform sensor deployment and divide the monitored area in coronas as illustrated in Figure 1a. A message transmitted from corona C_i is forwarded by sensor nodes in coronas C_{i-1}, C_{i-2}, and so on until it reaches corona C_1 from where it is transmitted to the sink. Corona width is chosen such that a message is forwarded by only one sensor in each corona. Assuming that each sensor is equally likely to be the source of a path to the sink, sensors suffer an uneven energy depletion, with sensors in the first corona being the first to die. This may result in network partitioning, with other sensors being unable to report their data to the sink.

Many papers covering the topic of sensor repositioning do not consider the issue of uneven energy depletion with distance to a predetermined sink; they are mainly concerned with uniformly distributing the sensors to provide load balancing and area coverage. In this paper, we consider a sink located in the center of the monitored area and propose algorithms to create a sensor movement plan that (1) achieves the desired sensor densities for uniform energy depletion (see Figure 1b), and (2) minimizes the distance that the sensors move.

The rest of this paper is organized as follows. Section II presents related works on sensor redeployment mechanisms. Section III shows the non-uniform sensor density computation and introduces the MSP problem definition. We continue in Section IV with three solutions for the MSP problem and their performance evaluation is presented in Section V. We conclude

TABLE I
PARTITIONING THE MONITORED AREA IN CORONAS, NOTATIONS.

C_i	The i^{th} corona
ρ_i	The computed, ideal density of corona C_i
A	Area of the whole monitored area
T	Data reporting interval
d	Width of each corona
n	Number of coronas
N	Total number of sensors
N_i	Number of sensors in corona C_i
R_c	Communication range of a sensor
R_s	Sensing range of a sensor

our paper in Section VI.

II. Related Works

The sensor placement issue has been widely studied recently [3], [7]. There are recent research works focusing on improving the initial deployment of WSNs using sensors' mobile ability [2], [11], [13]. Some research works (e.g. [2], [14]) provide centralized sensor deployment mechanisms, while others (e.g. [11], [13]) present distributed protocols.

In [2], Chellappan *et al.* study the flip-based deployment mechanism to achieve the maximum coverage. They assume the sensor can only flip once, and divide the whole network into multiple square regions. The centralized algorithm maximizes the number of regions that are covered by at least one sensor node with the minimum moving cost.

Wu and Yang introduce SMART [13], a scan-based distributed protocol with the goal of uniformly distributing sensors via sensor relocation. A scanning mechanism is used to balance the number of sensors first for each row of clusters and then for each column of clusters.

In [11], Wang *et al.* present VEC, VOR, and Minimax algorithms. To maximize the coverage, in VEC, sensors that are too close to each other will be pushed away by the virtual force. In VOR, when a node senses the coverage hole, it is moved towards the farthest vertex in the Voronoi polygon. The Minimax algorithm is similar to VOR; the virtual force will pull sensors to sparser areas. In [12], the authors proposed a grid-quorum solution to quickly locate the closest redundant sensors to the target area, where a sensor failure occurs.

In [14], Yang and Cardei use a partitioning of the monitoring area in coronas, and consider a one-time sensor flip mobility model to reposition sensors to prolong network lifetime while ensuring area coverage. The centralized sensor movement plan is computed by the sink using a max-flow min-cost approach.

Similar to [2], [11], [13], [14], we use sensor mobility to relocate them after the initial deployment. However, the goals in [2], [11], [13] involve improving the coverage and achieving load balance with uniform sensor densities. In [14], the goal is to achieve non-uniform sensor densities using a centralized algorithm when sensors can move by flipping at most once.

In this paper, we consider a general sensor movement model and propose both centralized and localized approaches that reposition sensors according to the computed non-uniform sensor densities. Using non-uniform sensor densities will prolong network lifetime by balancing sensors' energy depletion.

III. Non-uniform Sensor Density Computation and Problem Formulation

In this paper, we consider a general architecture where sensors send their measurements to a sink located centrally, as illustrated in Figure 1a. A WSN consisting of a large number of sensor nodes is deployed for periodic data reporting, where one data message per unit of area is transmitted each data-reporting-interval T. We consider a monitored area that is virtually divided in coronas, where the width of corona d equals the sensor communication range R_c. In this way, a message originating in corona C_i is forwarded by sensor nodes in coronas C_{i-1}, C_{i-2}, and so on until it reaches corona C_1 from where it is transmitted to the sink.

Table I shows the parameter notations used in corona partitioning. We assume that sensor energy consumption is proportional to the number of messages transmitted, and that sensors are uniformly deployed in the same corona. Intuitively, to balance energy consumption, we will deploy the fewest sensors in the last corona C_n and the largest number of sensors closest to the sink, in corona C_1. We define ρ_i to be the sensor density in the corona C_i, thus $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$.

Our objective is to compute the sensor density of each corona so that all sensors deplete their energy at the same rate. This will balance the energy consumption. For this, we require that each sensor transmits the same number of messages in the time interval T. We model the data gathering as a periodic data reporting process, where one data message is generated from each unit of area during the time interval T. Based on this assumption, a sensor in corona C_n will generate $1/\rho_n$ messages each time-interval T. Also note that sensors in C_n do not forward messages on behalf of other sensors.

Let us compute the number of messages $TotalNum_i$ transmitted by a sensor in corona C_i in time T. The sensor generates $1/\rho_i$ messages with its own measurements and participates in forwarding the messages generated by coronas $C_{i+1}, C_{i+2}, \ldots, C_n$. Using the notations in Table I, we have $N = \sum_{i=1}^{n} N_i$ and $A = \sum_{i=1}^{n} A_i$. The number of messages generated by all the sensors in corona C_i is N_i/ρ_i. Thus, the number of messages transmitted by a sensor in corona C_i in T time is:

$$TotalNum_i = \frac{1}{\rho_i} + \frac{\frac{N_{i+1}}{\rho_{i+1}} + \frac{N_{i+2}}{\rho_{i+2}} + \ldots \frac{N_n}{\rho_n}}{N_i} =$$

$$\frac{1}{\rho_i} + \frac{A_{i+1} + A_{i+2} \ldots + A_n}{A_i \cdot \rho_i} = \frac{A - (A_1 + A_2 + \ldots + A_{i-1})}{A_i \cdot \rho_i}$$

To balance the energy consumption, we require that sensors in different coronas consume the same energy, which means all sensors send the same number of messages in time T:

$$TotalNum_i = TotalNum_n$$

$$\frac{A - (A_1 + A_2 + \ldots + A_{i-1})}{A_i \cdot \rho_i} = \frac{1}{\rho_n}$$

$$\rho_i = \rho_n \cdot \frac{A - (A_1 + A_2 + ... + A_{i-1})}{A_i} \quad (1)$$

Value ρ_n can be computed based on the total number of sensors N and the coronas areas, as follows:

$$\begin{aligned} \rho_1 \cdot A_1 + \rho_2 \cdot A_2 + ... + \rho_n \cdot A_n &= N \\ \rho_n \cdot \frac{A}{A_1} \cdot A_1 + \rho_n \cdot \frac{A - A_1}{A_2} \cdot A_2 + ... & \\ + \rho_n \cdot \frac{A - A_1 - A_2 - ...A_{n-1}}{A_n} \cdot A_n &= N \end{aligned}$$

$$\Rightarrow \rho_n = \frac{N}{n \cdot A - (n-1) \cdot A_1 - (n-2) \cdot A_2 ... - A_{n-1}}$$

It follows that:

$$\rho_n = \frac{N}{A_1 + 2 \cdot A_2 ... + n \cdot A_n} = \frac{N}{\sum_{i=1}^{n} i \cdot A_i} \quad (2)$$

Combining equations (1) and (2), we can express the density ρ_i as a function of the total number of sensors, monitoring area, and coronas areas:

$$\rho_i = \frac{N}{\sum_{j=1}^{n} j \cdot A_j} \cdot \frac{A - \sum_{j=1}^{i-1} A_j}{A_i} \quad (3)$$

Considering n circular coronas of width d (see Figure 1a), $A_i = \pi \cdot d^2 (2 \cdot i - 1)$ and $A = \sum_{i=1}^{n} A_i = \pi (nd)^2$. After simple computations, equations (1) and (2) reduce to:

$$\rho_i = \rho_n \cdot \frac{n^2 - (i-1)^2}{2 \cdot i - 1} \quad and \quad \rho_n = \frac{N}{A} \cdot \frac{6n}{4n^2 + 3n - 1}$$

It follows that:

$$\rho_i = \frac{N}{A} \cdot \frac{6n}{4n^2 + 3n - 1} \cdot \frac{n^2 - (i-1)^2}{2 \cdot i - 1} \quad (4)$$

Next, we compute the improvement in network lifetime when using non-uniform densities versus the case of uniform sensor deployment. The case of uniform sensor deployment has been addressed in [8] and this work shows that the sensors in the first corona die first, thus limiting network lifetime.

Let us consider a uniform sensor distribution with density ρ. The total number of messages $TotalNum'_1$ transmitted by a sensor in corona C_1 in time T is:

$$TotalNum'_1 = \frac{1}{\rho} + \frac{\frac{N_2}{\rho} + \frac{N_3}{\rho} + \dots \frac{N_n}{\rho}}{N_1} = \frac{A}{N} \cdot n^2 \quad (5)$$

The improvement in network lifetime for a non-uniform sensor distribution compared to an uniform sensor distribution is:

$$\frac{TotalNum'_1}{TotalNum_n} = \frac{6n^3}{4n^2 + 3n - 1} \geq n \quad (6)$$

Equation (6) shows that by using a non-uniform sensor distribution we obtain a significant improvement in network lifetime, of at least n times.

Based on these computations, since the initial sensor deployment is random, our objective is to reposition sensors according to the densities computed in equation (3) while minimizing sensor movement. This will ensure a uniform energy depletion by the sensors in the network, maximizing the network lifetime. The sensor repositioning algorithm will be executed after the network deployment and before the data gathering protocol starts.

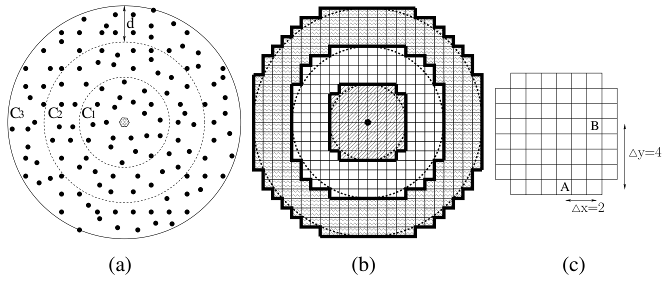


Fig. 2. Division of monitored area in $r \times r$ square regions. (a) Division of monitored area in coronas, (b) Division of monitored area in $r \times r$ square regions. (c) Manhattan distance between regions A and B.

The problem of Movement-assisted Sensor Positioning (MSP) is formalized as follows: *Given a WSN with N sensors randomly deployed for periodical monitoring of an area A centered to a sink, determine a sensor movement plan that will achieve sensor distribution in the monitored area according to equation (3), while minimizing the total sensor movement.*

IV. Solutions for the MSP Problem

In this section, we present three algorithms: a centralized Integer-Programming approach in Section IV-A, a localized matching algorithm in Section IV-B, and a distributed scan-based approach in Section IV-C. A discussion on providing the area coverage is presented in the Section IV-D.

A. Centralized Integer-Programming Approach

In modeling our problem, we divide the monitoring area into a grid of regions, where each region is an $r \times r$ square. Then, we divide the area in coronas, as represented in Figure 2b. For each region, let l be the smallest distance between a point in the region and the sink. Then the region belongs to corona C_i for $i = \lfloor l/d \rfloor$. In this case the division in coronas is not circular, but it follows the regions' contour. When the region's granularity is very small ($r \to 0$) the division in coronas is similar to the one in Figure 2a, where coronas are circular.

In this partitioning, we select d and r such that any node in corona C_i can directly reach corona C_{i-1}. This is satisfied if $R_c \geq d + r\sqrt{2}$. Our objective is to reposition sensors in order to achieve the desired density ρ_i in each corona C_i, according to the equation (3). This reduces to ensuring that each region in corona C_i achieves density ρ_i. Note that the equation (3) does not rely on circular corona partitioning, thus it applies to grid partitioning as well.

This section proposes a centralized approach. One solution is when the sink executes this algorithm. We consider that each region has a *representative* sensor which communicates with all the sensors in the region and with the sink. The representative determines the number of sensors in the region and transmits this information to the sink along with the region coordinates. The sink thus has a map of all the regions and the initial number of sensors in each region.

The sink computes the desired number of sensors in each region depending on the corona where the region resides. The desired number of sensors of a region in corona C_i is computed as $N_i^r = \rho_i \cdot r^2$. Note that N_i^r is a real number which is truncated to an integer $\lfloor N_i^r \rfloor$. The MSP problem is formulated as an Integer Programming (IP) that optimally determines the movement plan.

A region in some corona C_i ($i = 1 \ldots n$) can be a *source*, *hole*, or *neutral* region depending on whether the current number of sensors is greater than, less than, or equal to N_i^r. A bipartite graph $G = (V, U, E)$ is constructed where V, U are two node sets and E is the edge set. Source regions (hole regions) are represented as nodes in the set V (set U). Each node v has associated a weight $w(v)$, corresponding to the amount of sensor overload (if $v \in V$) or sensor underload (if $v \in U$).

We add edges between any two nodes in V and U. The weight of an edge is defined as the Manhattan distance between the corresponding source region and hole region. For example, in Figure 2c, the Manhattan distance between the regions A and B is $\Delta x + \Delta y = 2 + 4 = 6$.

We reduce the MSP problem to the problem of obtaining the desired densities N_i^r in each region using minimum sensor movement distance. Since the number of overloaded sensors is greater than or equal to the number of underloaded sensors (due to the rounding of the N_i^r values), this problem reduces to matching all underloaded regions such that the sum of the weights of the selected edges is minimized.

We define x_{ij} where $i = 1 \ldots |U|$, $j = 1 \ldots |V|$, and $x_{ij} \in \{0, 1, \ldots, min(w(v_i), w(u_j))\}$ as the number of sensors that will move from the source region v_i to the hole region u_j. We denote c_{ij} as the weight of the edge (v_i, u_j). The optimal solution is defined using IP-formulation:

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{ij} c_{ij} x_{ij} \\ &\text{subject to} && \textstyle\sum_{j=1}^{|V|} x_{ij} \leq w(v_i) \text{ for all } i = 1 \ldots |U| \\ & && \textstyle\sum_{i=1}^{|U|} x_{ij} = w(u_j) \text{ for all } j = 1 \ldots |V| \end{aligned}$$

Remarks:

- The objective function asks to minimize the total sensor moving distance.
- The first constraint requires that the number of sensors that leave the source region v_i be upperbounded by $w(v_i)$, which is the overload of that region.
- The second constraint requires that the number of sensors that enter a hole region u_j be $w(u_j)$, which is the underload of that region.

The sink uses an IP-solver to compute the sensor movement plan (given by the x_{ij} values) and forwards it to the region representatives which coordinate the senor movement inside that region. The IP has a large running time for a large number of variables. We reduce the IP to the assignment problem, also known as the Hungarian method [9], which can be solved on $O(m^3)$ time for m variables.

We transform the bipartite graph G to a bipartite graph $G' = (V', U', E')$ as follows. V' contains the overloaded sensors from all the source regions, and U' the underloaded sensors from all the hole regions. Since $|V'| \geq |U'|$, we add $|V'| - |U'|$ virtual nodes u^* in U'. Set E' contains an edge between any two nodes in V' and U' with weight defined as the Manhattan distance between the regions of the two sensors. Edges joining a node u^* have weight 0.

We define $x_{ij} = 1$ if edge (v_i, u_j) is selected in the matching and $x_{ij} = 0$ otherwise. c_{ij} denotes the weight of the edge (v_i, u_j). Then the 0-1 IP respects the general form of the assignment problem:

$$\begin{aligned} &\text{Minimize} && \textstyle\sum_{ij} c_{ij} x_{ij} \\ &\text{subject to} && \textstyle\sum_{j=1}^{|V'|} x_{ij} = 1 \text{ for all } i = 1 \ldots |U'| \\ & && \textstyle\sum_{i=1}^{|U'|} x_{ij} = 1 \text{ for all } j = 1 \ldots |V'| \end{aligned}$$

This assignment problem is solvable [9] in $O(m^3)$ time for $m = |V'|^2$ variables. Note that the virtual nodes u^* do not participate in the movement plan and they contribute a cost of 0 to the objective function. We use CPLEX solver to implement the IP. Simulation results are presented in Section V.

B. Localized Matching Method

In this section, we extend the solution from Section IV-A to a localized approach. Similar to Section IV-A, we consider a division of the monitored area in an $r \times r$ grid of square regions, see Figure 2b. To ensure that a sensor in a region can directly communicate with any sensor in an adjacent region (left, right, top, or bottom), we choose r such that $r \leq R_c/\sqrt{5}$.

Each region selects a *representative* in charge of communication with the neighbor regions' representatives and with organizing the movement inside the region. This corresponds to a movement model where sensors in a region can move only to the neighbor regions: left, right, top, and bottom. Thus, the movement distance between two regions is computed as the Manhattan distance.

According to the classification in Section IV-A, a region in corona C_i can be a *source*, *hole*, or *neutral* region depending on whether the current number of sensors is greater than, less than, or equal to N_i^r. Let us denote Δ_+ as the number of overloaded sensors in a source region and Δ_- as the number of underloaded sensors in a hole region.

The movement protocol is initiated by the hole regions and is a three-way message exchange protocol. Since the total number of overloaded sensors is greater than or equal to the number of underloaded sensors, all hole region requirements will be satisfied when the algorithm completes. The main steps of the algorithm executed by the source and hole regions are summarized using pseudo-code.

Algorithm 1 Localized Matching Method - Hole Region

```
1: determine the underloaded value $\Delta_-$ and wait a random delay
2: broadcast *Request* message including $\Delta_-$; use TTL to limit the
   number of hops
3: **if** *Reply* messages received **then**
4:     compute movement plan including the number of sensors to
       be moved from each source; give priority to the closer sources.
5:     broadcast *MovementPlan* using TTL mechanism to limit the
       number of hops
6: **end if**
7: after the movement phase, update $\Delta_-$
8: **if** $\Delta_- > 0$ **then**
9:     $TTL \leftarrow TTL + \delta$
10:    goto line 2
11: **else if** $\Delta_- = 0$ **then**
12:    change status to *neutral* region
13: **end if**
```

Algorithm 2 Localized Matching Method - Source Region

```
1: determine the overload value $\Delta_+$
2: **if** *Request* message received **then**
3:     reserve $min(\Delta_+, \Delta_-)$ sensors for some specific time
4:     send back *Reply* message with the number of sensors
       $min(\Delta_+, \Delta_-)$ allocated for this request
5: **end if**
6: **if** *MovementPlan* message received that requests $n^*$ sensors from
   this source **then**
7:     move $n^*$ sensors to the hole region
8:     update $\Delta_+ \leftarrow \Delta_+ - n^*$
9:     **if** $\Delta_+ = 0$ **then**
10:        change status to *neutral* region
11:    **end if**
12: **end if**
```

A hole region waits for a random amount of time and then broadcasts a *Request* message including the underload Δ_- and a TTL (Time-To-Live). All intermediate regions that receive the message for the first time decrease the TTL by 1 and forward the message. The TTL is used to control the number of hops that a message is forwarded.

Besides participating in data forwarding, a source region receiving a *Request* message sends back a *Reply* message containing the number of sensors $min(\Delta_-, \Delta_+)$ it allocates and reserves for this request. This is a unicast message transmitted back to the hole region that initiated the request.

Once the hole region receives the *Reply* messages, it computes the movement plan, specifying, for each source region, the number of sensors it has to move. If the number of sensors reserved by the sources is less than or equal to Δ_-, then all of them are included in the movement plan. If the number of sensors reserved by the sources is greater than Δ_-, then the sensors from the closer source regions are added in the movement plan first. This selection criteria helps to minimize the sensors movement distance. The hole then broadcasts a message *MovementPlan* with the same TTL value used in the *Request* message. All intermediate regions that receive the message for the first time decrease the TTL by 1 and forward the message.

The actual sensor movement takes place when a source region receives a *MovementPlan* message. After the sensor movement, the source region updates Δ_+.

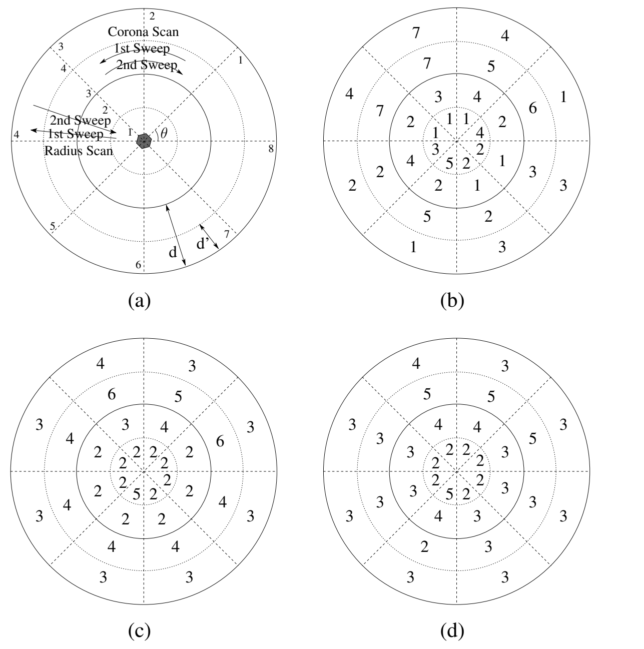


Fig. 3. An example for the scan-based approach (a) Partitioning of the monitored area in 4 rings and 8 sectors, (b) Initial sensor deployment, (c) Sensor deployment after the corona scan, (d) Sensor deployment after the radius scan.

There may be cases when not all of the underloaded sensors are filled in the first iteration. In this event, the process is repeated using an expanding ring search mechanism. Thus, in the next iteration the search is performed using $TTL = TTL + \delta$, where δ is a predefined constant. The whole matching process terminates when all of the hole regions have filled out their underload values and thus have become neutral regions.

C. Scan-based Approach

In this section, we present a distributed approach using a corona-radius scanning mechanism. We consider the network to be virtually partitioned in coronas and sectors, as represented in Figure 3a. Initially, we consider a virtual division of the monitoring area in coronas (see Figure 1a) with width d.

To further control the sensor distribution and to ensure communication to adjacent regions, we consider a partitioning into thinner coronas (or *rings*) of width d' (where $d' = d/\xi$, for some integer ξ) and sectors with angle θ as shown in Figure 3a. Angle θ is chosen such that sensors in a region can communicate with sensors in the left and right regions in the same coronas. Values θ and ξ determine the region granularity and therefore the total monitoring area is divided into $n \cdot \frac{d}{d'} \cdot \frac{360^\circ}{\theta}$ regions.

The desired density of a region depends on the corona where that region belongs and is computed according to equation (3). Figures 3b and 3d show an example with the number of sensors in each region after the initial deployment and the desired number of sensors in each region, respectively.

We assume that sensors are densely deployed such that each region has at least one sensor. Each region i has a *representative* in charge of communication with the adjacent regions' representatives, and has the following information: (1) region i's position in the currently processed corona/sector, and (2) the number of sensors w_i in the region.

Two scans are used in sequence: *corona scans* followed by *radius scans*. A corona scan will balance the number of sensors per corona and at the end of this scan, regions in the same corona will have the same number of sensors, see Figure 3c. In the second scan (radius scan), sensors are redistributed on segments according to the desired sensor densities, see Figure 3d. Each scan has two sweeps, which are described next.

Corona Scan. The first sweep scans the regions from region 1 to region $t = \frac{360^\circ}{\theta}$ numbered as in Figure 3a, and the second sweep in the reverse direction, from region t to region 1. During the first sweep, each region i determines the number of sensors w_i in the region, computes the prefix sum $v_i = v_{i-1} + w_i$, and forwards v_i to the next region. The last region computes v_t and $\overline{w} = \lfloor v_t/t \rfloor$ and initiates the second sweep by propagating back $\overline{w}$. As a result of the second sweep, all regions have at least $\overline{w}$ sensors. Some regions might have more sensors since v_t/t is a real number truncated to an integer. The second sweep is illustrated using pseudo-code.

Algorithm 3 Corona Scan - Second Sweep (region i)

```
1:  if $i = t$ OR $Balanced(\overline{w})$ message received then
2:      go to line 11
3:  else if $RequestSensors(\overline{w}, m)$ message received then
4:      $w_i \leftarrow w_i - m$
5:      $v_i \leftarrow v_i - m$
6:      move $m$ sensors to region $(i+1)$
7:  else if $MoveSensors(\overline{w}, m)$ message received then
8:      $w_i \leftarrow w_i + m$
9:      $v_i \leftarrow v_i + m$
10: end if
11: if $i = 1$ then return
12: $\overline{v}_i \leftarrow i \cdot \overline{w}$
13: if $w_i = \overline{w}$ then
14:     send $Balanced(\overline{w})$ message to region $(i-1)$
15: else if $w_i > \overline{w}$ then
16:     if $w_i - \overline{w} > v_i - \overline{v}_i$ then
17:         $m \leftarrow w_i - \overline{w} - (v_i - \overline{v}_i)$
18:         send $MoveSensors(\overline{w}, m)$ message to region $(i-1)$
19:         move $m$ sensors to region $(i-1)$
20:     else
21:         send $Balanced(\overline{w})$ message to region $(i-1)$
22:     end if
23: else if $w_i < \overline{w}$ then
24:     send $RequestSensors(\overline{w} - w_i)$ message to region $(i-1)$
25: end if
```

During the second sweep, the representative of each region i receives $\overline{w}$ from the region $i+1$ and computes $\overline{v}_i = \lfloor i \cdot \overline{w} \rfloor$. The representative of each region i, for $1 < i < t$, receives one message ($Balanced$, $RequestSensors$, or $MoveSensors$) from the upstream region $i+1$ and sends one message ($Balanced$, $RequestSensors$, or $MoveSensors$) to the downstream region $i-1$. Exceptions are the region t (which is the initiator of the sweep and thus it does not receive a message), and region 1 (which ends the sweep process and thus does not issue any message).

A region i updates (see lines $1 \dots 10$) the values w_i and v_i depending on the type of message received from the region $i+1$ and sends region $i-1$ one of the three messages as follows:

- If $w_i = \overline{w}$, then the state of this region is neutral, and thus it does not have to receive/send any sensors.
- If $w_i > \overline{w}$, then this is a source region. The region sends sensors to region $i-1$ only if the downstream regions need additional sensors, that means $\overline{v}_{i-1} > v_{i-1}$ which is equivalent to $w_i - \overline{w} > v_i - \overline{v}_i$. In this case, the number of sensors to be sent to region $i-1$ is $\overline{v}_{i-1} - v_{i-1} = w_i - \overline{w} - (v_i - \overline{v}_i)$, and a message *MoveSensors* is transmitted. Otherwise the region representative transmits a *Balanced* message, used to propagate the value $\overline{w}$.
- If $w_i > \overline{w}$, then this is a hole region. The representative of this region requires additional $\overline{w} - w_i$ sensors from the region $i-1$ using *RequestSensors* message.

Note that in lines 6 and 19, sensor movement takes place when sensors become available in that region. There are cases when the region has to receive sensors before forwarding. At the end of this scan, regions in the same corona ring have at least $\overline{w}$ sensors (which is the average value taken over all the sensors in the same ring). There may be regions with more sensors since the average value v_t/t was truncated to an integer. If v_t/t is an integer, then all regions will have the same number of sensors, see Figure 3c.

Radius Scan. Let us denote the regions in a sector from 1 to p, where $p = n \cdot \frac{d}{d'}$, with region 1 being the region closest to the sink, see the notations in Figure 3a. Two sweeps take place, one from region 1 to region p and another in the reverse direction from region p to region 1. We denote w_i as the number of sensors in region i. During the first sweep, each region i computes the prefix sum $v_i = v_{i-1} + w_i$ and forwards v_i to the next region. The last region computes v_p and initiates the second sweep by propagating back v_p.

Compared to corona scanning, sensor distribution is not uniform; it has different densities depending on the distance to the sink. In the second sweep, the representative of each region i computes the area of region i, $Area_i = d'^2(2i-1)\frac{\theta}{2}$. The desired (or target) number of sensors of region i is computed as $t_i = \lfloor \rho_k \cdot Area_i \rfloor$, where ρ_k is the density of the corona $k = \lceil i \cdot \frac{d'}{d} \rceil$. The density ρ_k is computed using equation (3) with $N = v_p \cdot \frac{360^\circ}{\theta}$ and $A = Area_{sector} \cdot \frac{360^\circ}{\theta}$, where $Area_{sector} = (nd)^2 \cdot \theta/2$. In addition, the representative of region i computes the desired number of sensors $t_1, t_2, \dots, t_{i-1}$ and $\overline{v}_i = \sum_{j=1}^{i} t_j$.

The second sweep is illustrated using pseudo-code and its description is similar to that of the corona sweep. As a result of executing the second sweep, each region i will have at least t_i sensors. Some regions might result in more sensors since t_i was truncated to an integer.

Figure 3 shows an example for the scan-based approach.

Algorithm 4 Radius Scan - Second Sweep (region i)

```
1: if $i = t$ OR $Balanced(v_p)$ message received then
2:     go to line 11
3: else if $RequestSensors(v_p, m)$ message received then
4:     $w_i \leftarrow w_i - m$
5:     $v_i \leftarrow v_i - m$
6:     move $m$ sensors to region $(i + 1)$
7: else if $MoveSensors(v_p, m)$ message received then
8:     $w_i \leftarrow w_i + m$
9:     $v_i \leftarrow v_i + m$
10: end if
11: if $i = 1$ then return
12: compute $t_i$, $\overline{v}_i$
13: if $w_i = t_i$ then
14:     send $Balanced(v_p)$ to region $(i - 1)$
15: else if $w_i > t_i$ then
16:     if $w_i - t_i > v_i - \overline{v}_i$ then
17:         $m \leftarrow w_i - t_i - (v_i - \overline{v}_i)$
18:         send $MoveSensors(v_p, m)$ message to region $(i - 1)$
19:         move $m$ sensors to region $(i - 1)$
20:     else
21:         send $Balanced(v_p)$ to region $(i - 1)$
22:     end if
23: else if $w_i < t_i$ then
24:     send $RequestSensors(t_i - w_i)$ to region $(i - 1)$
25: end if
```

In Figure 3a, the monitored area with the sink in the center is divided in 8 sectors and each corona (illustrated with continuous circles) is divided in 2 rings. In the corona scan, the regions in the same ring are labeled from 1 to 8 in the counterclockwise direction. In the radius scan, the regions in the same sector are labeled from 1 to 4 starting with the inner region as shown in Figure 3a.

Figure 3b shows an initial deployment of 100 sensors. The number in each region shows the initial number of sensors. Figures 3c and 3d show the number of sensors in each region after the corona scan and the radius scan, respectively. After both scans, each region gets the desired number of sensors according to their different density requirements.

D. *Discussion on the Area Coverage*

The main design criteria considered in this paper is improving network lifetime using sensor repositioning. Another important objective in many applications is ensuring area coverage.

In all three of the solutions proposed, the monitored area is divided into smaller regions. In the first two algorithms, we use an $r \times r$ grid partition. Assuming the area covered by a sensor to be a disk with radius R_s, area coverage can be guaranteed if we select r such that $R_s \geq r\sqrt{2}$. Then the existence of a sensor in a region ensures the coverage of that region. In the scanning-based algorithm we use a different area partitioning, and similarly we can ensure area coverage if we choose the parameters d' and θ such that any region is covered by a sensor with sensing range R_s.

When the region size is larger and it cannot be covered by only a sensor in the region, a hierarchical approach can be developed. First, one of the three proposed algorithms is used to position sensors in regions to satisfy the density requirements from the equation (3). Second, a region can be partitioned in smaller regions such that a sensor in a smaller region guarantees its coverage. Algorithms that uniformly balance the number of sensors among smaller regions have been proposed in literature [13].

V. Simulation Results

In this section, we present simulation results of our solutions to the MSP problem: Integer Programming (IP), localized matching method, and scan-based approach.

A. *Simulation environment*

Metrics in the simulation are used to measure the deployment quality and the cost of the algorithms. Deployment quality is represented by the network lifetime. The total moving distance and the overhead are used to measure the cost of the algorithms. The number of iterations is also examined for the localized matching method.

Network lifetime is defined as the number of rounds the network lasts before the first sensor runs out of energy. Each unit of area generates one message every round. In our simulations, we account the energy consumed for message transmissions and consider that e represents the energy consumed per message, where $e = 1$ unit. Each sensor has the total energy $E = 5000$ units. The total number of rounds for each sensor i is calculated as $\frac{E_i}{M_i \cdot e}$, where M_i is the total number of messages sensor i transmits. Network lifetime is computed as the minimum number of rounds.

The total moving distance is defined as $\sum_{ij} c_{ij} x_{ij}$, where x_{ij} is the number of sensors that have moved from source region i to hole region j. In the IP and the localized matching methods, c_{ij} is the Manhattan distance between regions i and j. In the scan-based approach, c_{ij} is the average distance between regions i and j. In $CoronaScan$, the average distance in ring i is computed as $\frac{2 \cdot \pi \cdot (i - \frac{1}{2}) \cdot ringWidth}{sectorNum}$, where $ringWidth$ is the width of the ring and $sectorNum$ is the number of sectors. In $RadiusScan$, the average distance is always the width of the ring.

The overhead of the algorithm is defined as the total number of messages exchanged between source regions and hole regions. Since the localized matching method is conducted iteratively, the number of iterations is also taken into account.

We conduct the simulation on a custom discrete event simulator, which generates the random initial sensor deployment. All the tests are repeated 200 times. In the simulation, we use the following variable parameters:

- The diameter of the monitored area disk is 360 units.
- The total number of sensors in the network varies from 1500 to 2500.
- The corona width is 60 units.
- The region size varies from 15 to 60 units.
- Three expansion speeds of TTL (ΔTTL). These are $\Delta TTL = 1$ means TTL increases linearly with step 1, then TTL = 1, 2, 3, etc. $\Delta TTL = 3$ means TTL increases linearly with step 3, then TTL = 1, 4, 7, etc. $\Delta TTL = 0$

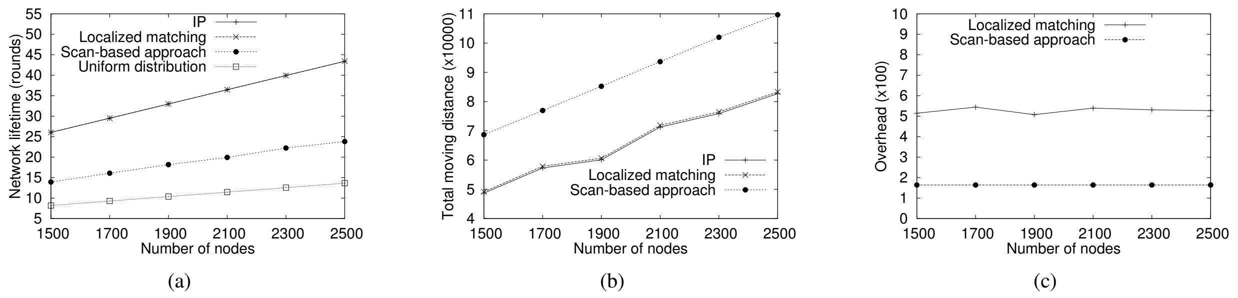


Fig. 4. Comparison among IP approach, localized matching method and scan-based approach. (a) Network lifetime comparison, (b) Total moving distance comparison, (c) Overhead comparison.

means TTL increases exponentially with base 3, TTL = 1, 3, 9, etc.

- The number of sectors varies from 8 to 32.
- The width of the ring varies from 20 to 60 units.

B. Simulation results

In Figure 4, the region size in the IP and the localized matching methods is 30×30 units. In the scan-based approach, the width of the ring is 30 units and the number of sectors is 8. Figure 4a compares the network lifetime among the distributions after executing our three algorithms and the uniform distribution. IP and localized matching methods have similar performance and they are better than the distribution achieved by the scan-based approach and uniform distribution. There are three coronas in the monitored area and the network lifetime achieved by the localized matching method is more than three times larger than that achieved by the uniform distribution. This is coherent with the analysis in Section III.

Network lifetime achieved by the scan-based approach is smaller than that achieved by the localized matching method for the following reason. In the scan-based approach, when conducting the $CoronaScan$, the actual average number of sensors in each region is rounded into the floor of the exact average real number, and then during the $RadiusScan$, the target number in each region is computed according to the actual number v_p in this sector. When we use equation (3) to compute the target density of a corona with $N = v_p \cdot \frac{360^\circ}{\theta}$, N is less than the actual total number of sensors in the monitored area. Therefore, some regions may have fewer sensors and consequently they become the bottleneck, which leads to a shorter network lifetime.

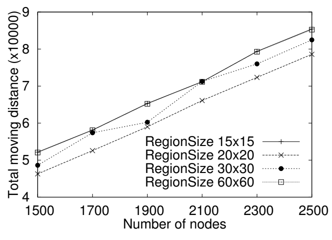
Figure 4b compares the total moving distance. The localized matching method gets close results to that of the IP approach, which is the optimal solution. There are two reasons that the total moving distance of the scan-based approach is larger than that in the localized matching method. First, the sensor movement in the $CoronaScan$ might not be optimal. Consider the worst case when there is only one source region t and one hole region 1. Then according to our approach, sensors move through regions $t, t-1, t-2, \ldots, 1$, when the optimal way is to move them directly from region t to region 1. Second, $CoronaScan$ introduces an additional step that moves sensors to balance the number of sensors in coronas. Some of this sensors movement might be avoided in an optimal movement plan.

Figure 4c shows the overhead of the localized matching method and the scan-based approach. The localized matching method is executed iteratively and it involves three-way message exchange in each iteration. Thus, although it gets shorter moving distance, it suffers from higher overhead compared to the scan-based approach. A trade-off between the total moving distance and the overhead exists when comparing these two algorithms.

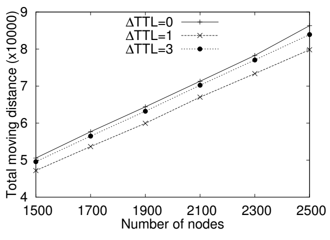
Figure 5 studies the influence of region sizes and TTL expansion speeds on the performance of the localized matching method. Figure 5a shows the total moving distances under different region sizes when $\Delta TTL = 1$. When region sizes are 15×15 and 60×60, the method has a greater moving distance. This is because as the region size decreases, the number of movements increases. As the region size increases, the number of movements decreases but the distance between two regions is larger. In our simulation environment, a region size of 20×20 achieves the shortest total moving distance.

In Figures 5b, 5c, and 5d, the region size is 20×20. In Figure 5b, $\Delta TTL = 0$ has the largest moving distance. This is because when TTL has a fast increase, source regions reserve sensors first for hole regions with a shorter initial waiting times. Increasing the TTL dramatically may cause more suboptimal matches, which means hole regions match with source regions farther away.

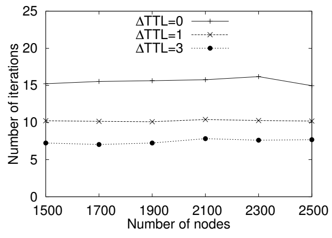
In Figures 5c and 5d, we compare the number of iterations and the overhead for various TTL expansion speeds. Our simulation results show that using $\Delta TTL = 1$ requires more iterations but produces less overhead. When $\Delta TTL = 0$, the number of iterations and overhead are larger than the two other TTL cases. This is because when TTL increases linearly, the matched pairs of source and hole regions are usually resolved for smaller ranges. When the TTL increases exponentially, a hole region with the smallest delay may reserve sensors in many source regions, causing other hole regions to unfulfill their requirements and thus to have to wait for matching in the following iterations. Thus the number of iterations increases in this case.



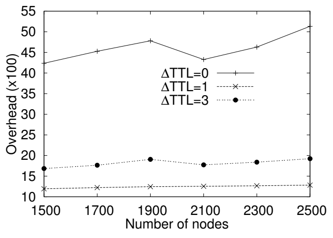
(a)



(b)

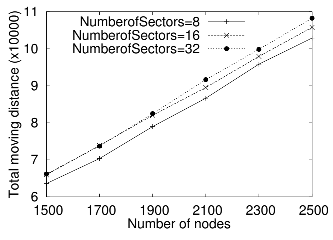


(c)

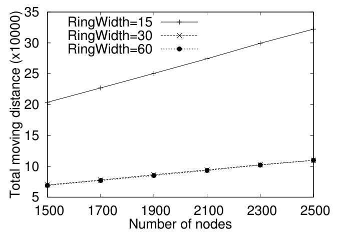


(d)

Fig. 5. Comparison for the localized matching method.



(a)



(b)

Fig. 6. Comparison for the scan-based approach.

In Figure 5d, $\Delta TTL = 0$ produces more overhead than the two other cases since it uses more iterations. $\Delta TTL = 3$ produces more overhead than $\Delta TTL = 1$ since with a larger increase in the TTL more source regions receive *Request* messages sent by hole regions and then send back *Reply* messages, sometimes resulting in more reservations than the actual number of sensors requested.

Figure 6 shows the effect of the number of sectors and rings on the total moving distance in the Scan-based approach. In the Figure 6a, the ring width is 60 units and the shortest moving distance is obtained when the number of sectors is 8. This is because with the increase in the number of regions, the number of movements increases too, resulting in the increase of the total moving distance. In Figure 6b, the number of sectors is 8 and the shortest moving distance is obtained when the ring width is 60 units. With the increase of the number of rings, the number of movements in the $CoronaScan$ is increased, resulting in a larger total moving distance.

Simulation results can be summarized as follows: (1) Using the localized matching method and the scan-based approach, the network lifetime is effectively prolonged compared with the uniform distribution, (2) The localized matching method has shorter total moving distance but larger overhead compared to the scan-based approach, (3) Region size must be selected carefully since it affects the algorithm performance, (4) Linear expansion speed of TTL has better effect on the performance of the localized matching method.

VI. Conclusions and Future Work

In this paper we focus on sensor redeployment that will prolong network lifetime while minimizing sensor movement. With the observation that sensors closer to the sink tend to consume more energy than those farther away from the sink, we compute the desired non-uniform sensor densities based on the distance to the sink. We propose three algorithms to reposition sensors according to these densities: an IP-based mechanism that produces the optimal solution, a localized matching algorithm which is scalable with large WSNs, and a low-overhead distributed scanning-based mechanism.

In our future work, we plan to study sensor distribution and repositioning for other data gathering models such as event-based data gathering.

Acknowledgments

This work was supported in part by NSF grants CCF 0545488, CNS 0422762, CNS 521410, CCR 0329741, CNS 0434533, CNS 0531410, and CNS 0626240.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, Wireless sensor networks: a survey, *Computer Networks*, 38(4), pp. 393-422, 2002.

[2] S. Chellappan, X. Bai, B. Ma, D. Xuan, and C. Xu, Mobility Limited Flip-Based Sensor Networks Deployment, *IEEE Transactions of Parallel and Distributed Systems*, 18(2), pp. 199-211, 2007.

[3] T. Clouqueur, V. Phipatanasuphorn, and K. K. Saluja, Sensor deployment strategy for target detection, *WSNA*, 2002.

[4] K. Dantu, M. H. Rahimi, H. Shah, S. Babel, A. Dhariwal, G. S. Sukhatme, Robomote: enabling mobility in sensor networks, *IPSN 2005*, pp. 404-409, 2005.

[5] 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*, Sept. 2002.

[6] U. Lee, E.O. Magistretti, B.O. Zhou, M. Gerla, P. Bellavista, and A. Corradi, Efficient data harvesting in mobile sensor Platforms, *PerCom Workshops*, pp. 352-356, 2006.

[7] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak, Exposure in wireless ad-hoc sensor networks, *Mobicom*, 2002.

[8] S. Olariu and I. Stojmenovic, Design guidelines for maximizing lifetime and avoiding energy holes in sensor networks with uniform distribution and uniform reporting, *IEEE INFOCOM*, 2006.

[9] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization, algorithms, and complexity*, Dover publications, INC, 1998.

[10] J. Tan, O. M. Lozano, N. Xi, and W. Sheng, Multiple Vehicle Systems for Sensor Network Area Coverage, *5th World Congress on Intelligent Control and Automation*, 2004.

[11] G. Wang, G. Cao, and T. F. LaPorta, Movement-assisted sensor deployment, *IEEE Transactions on Mobile Computing*, 5(6), pp. 640-652, 2006.

[12] G. Wang, G. Cao, T. F. LaPorta, and W. Zhang, Sensor relocation in mobile sensor networks, *INFOCOM'05*, 2005.

[13] J. Wu and S. Yang, SMART: a scan-based movement-assisted sensor deployment method in wireless sensor networks, *INFOCOM'05*, pp. 2313-2324, 2005.

[14] Y. Yang and M. Cardei, Movement-Assisted Sensor Redeployment Scheme for Network Lifetime Increase, *MSWIM'07*, Oct. 2007.