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

Good sensor deployment is vital for wireless sensor networks (WSNs). To improve the initial deployment and to prolong network lifetime, one approach is to relocate 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. 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: an Integer-Programming formulation, a localized matching method, and a distributed corona-radius scanning algorithm. Simulation results are presented to evaluate the proposed solutions.

1 Introduction and Related Works

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 collected data to sinks. 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 [3], springs [2], or they can be attached to transporters such as robots [3] and vehicles [5].

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 in this case. 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 to improve the initial deployment: deploying additional sensors [4] or movement-assisted sensor positioning mechanisms [2, 9, 10].

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

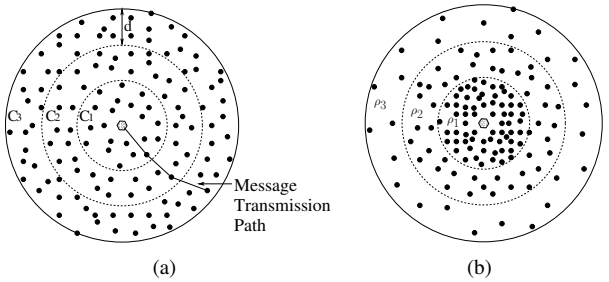

Figure 1. WSN model using coronas concentric to the sink. (a) Uniform distribution, (b) Non-uniform distribution.

In a WSN, sensors closer to the sink tend to consume more energy than those farther away from the sink. In [6], Olariu *et al.* consider a uniform sensor deployment and divide the monitored area in coronas (see 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 is because, besides transmitting their own packets, they forward packets on behalf of other sensors. This may result in network partitioning and reduction of network lifetime, 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 propose algo-

978-1-4244-2100-8/08/$ 25.00

Table 1. Notations.

C_i	The i^{th} corona
ρ_i	The computed, ideal density of corona C_i
A	Area of the whole monitored area
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

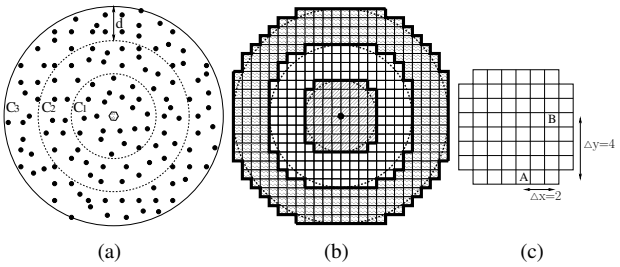

Figure 2. Division of monitored area. (a) in coronas, (b) in $r \times r$ square regions. (c) Manhattan distance between two regions.

rithms 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.

There are recent research works [2, 10, 9] focusing on improving the initial deployment of WSNs using sensors' mobile ability. 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 [9], 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 [10], Yang and Cardei consider a one-time sensor flip mobility model to reposition sensors. The movement plan is computed by the sink using a max-flow min-cost approach.

The goals in [2, 9] involve improving the coverage and achieving load balance with uniform sensor densities. In [10], the goal is to achieve non-uniform sensor densities using a centralized algorithm when sensors can move by flipping at most once.

2 Problem Formulation

Similar to [6] and [10], 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 period. 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 1 shows the parameter notations used in corona partitioning.

Paper [10] computes the sensor density ρ_i for each corona C_i such that all sensors deplete their energy at the same rate:

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

The initial sensor deployment is random, and our goal is to reposition sensors according to the densities computed in equation (1) 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.

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 (1), while minimizing the total sensor movement*.

3 Solutions for the MSP Problem

3.1 Integer Programming Approach

We divide the monitoring area into $r \times r$ square regions and then coronas (see 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 ($r \to 0$), it is similar to the one in Figure 2a, where coronas are circular.

In this partitioning, we select d and r, $R_c \geq d + r\sqrt{2}$, such that any node in corona C_i can directly reach corona

C_{i-1}. Our objective is to reposition sensors in order to achieve the desired density ρ_i in each corona C_i, according to the equation (1). This reduces to ensuring that each region in corona C_i achieves density ρ_i. Note that the equation (1) does not rely on circular corona partitioning, thus it applies to grid partitioning as well.

This section proposes a centralized approach, which can be executed by the sink. We consider that each region has a *representative* sensor which communicates with all the sensors in the region and with the sink. It determines the number of sensors in the region and transmits this information to the sink. 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 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. 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 $\Delta x + \Delta y = 2 + 4 = 6$.

The goal of the MSP problem is to obtain the desired densities N_i^r in each region using minimum 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:

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

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 [7], 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:

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

This assignment problem is solvable [7] 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 4.

3.2 Localized Matching Method

In this section, we extend the solution from Section 3.1 to a localized approach. Similar to Section 3.1, 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 $r \leq R_c/\sqrt{5}$.

Each region selects a *representative* in charge of communication with the neighbor regions' representatives and with organizing the movement inside the region. Sensors in

Algorithm 1 Localized Matching Method - Hole Region

1: determine the underloaded value Δ_- and wait a random delay
2: broadcast *Request* message including Δ_-; 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 Δ_-
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 Δ_+
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 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.

Similar to Section 3.1, a region in corona C_i can be *source*, *hole*, or *neutral* region if the current number of sensors is greater than, less than, or equal to N_i^r. Δ_+ is the number of overloaded sensors in a source region and Δ_- is 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.

A hole region waits 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 it. TTL is used to control the number of hops that a message is forwarded.

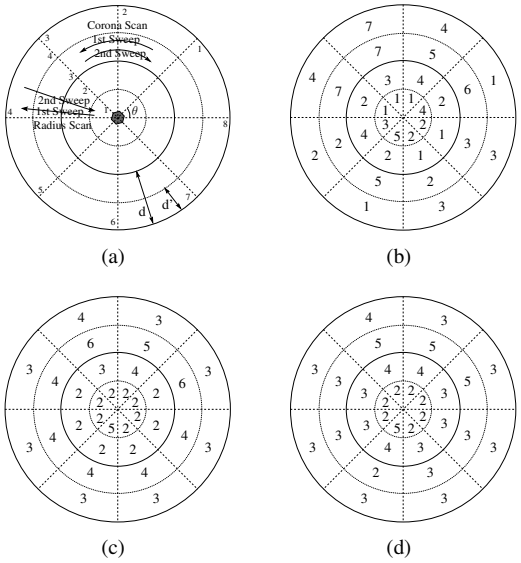

Figure 3. Example for the scan-based approach (a) Area partitioning, (b) Initial deployment, (c) Deployment after the corona scan, (d) Deployment after the radius scan.

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 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 it.

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

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.

3.3 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 (see 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 (1). Figures 3b and 3d show an example with the initial deployment and the desired number of sensors in each region, respectively.

We assume that sensors are densely deployed and 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, (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 radius scan, sensors are redistributed on segments according to the desired sensor densities, see Figure 3d. Each scan has two sweeps 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 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 $\overline{w}$. After 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.

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).

Algorithm 3 Corona Scan - Second Sweep (region i)

```
1:  if i = t OR Balanced(w̄) message received then
2:     go to line 11
3:  else if RequestSensors(w̄, m) message received then
4:     w_i ← w_i − m
5:     v_i ← v_i − m
6:     move m sensors to region (i + 1)
7:  else if MoveSensors(w̄, m) message received then
8:     w_i ← w_i + m
9:     v_i ← v_i + m
10: end if
11: if i = 1 then return
12: v̄_i ← i · w̄
13: if w_i = w̄ then
14:     send Balanced(w̄) message to region (i − 1)
15: else if w_i > w̄ then
16:     if w_i − w̄ > v_i − v̄_i then
17:         m ← w_i − w̄ − (v_i − v̄_i)
18:         send MoveSensors(w̄, m) message to region (i − 1)
19:         move m sensors to region (i − 1)
20:     else
21:         send Balanced(w̄) message to region (i − 1)
22:     end if
23: else if w_i < w̄ then
24:     send RequestSensors(w̄, w̄ − w_i) message to region (i −
        1)
25: end if
```

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

• 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 forward-

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 $RequestSensors(v_p, t_i - w_i)$ to region $(i-1)$
25: **end if**

ing. 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 (1) 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 $t_1, t_2, \ldots, t_{i-1}$ 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. After 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. 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.

4 Simulations

Simulation environment. Metrics in the simulation include the network lifetime, the total moving distance, and the overhead. 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 dies. Each unit of area generates one message every round. We account the energy consumed for message transmissions and consider $e = 1$ unit representing the energy consumed per message. Each sensor has total energy $E = 5000$ units. The total number of rounds for each sensor i is $\frac{E_i}{M_i \cdot e}$, where M_i is the total number of messages sensor i transmits.

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 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 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 simulations on a custom discrete event simulator, which generates the random initial sensor deployment. All the tests are repeated 200 times. In the simulation, the diameter of the monitored area disk is 360 units. The corona width is 60 units. And there are three expansion

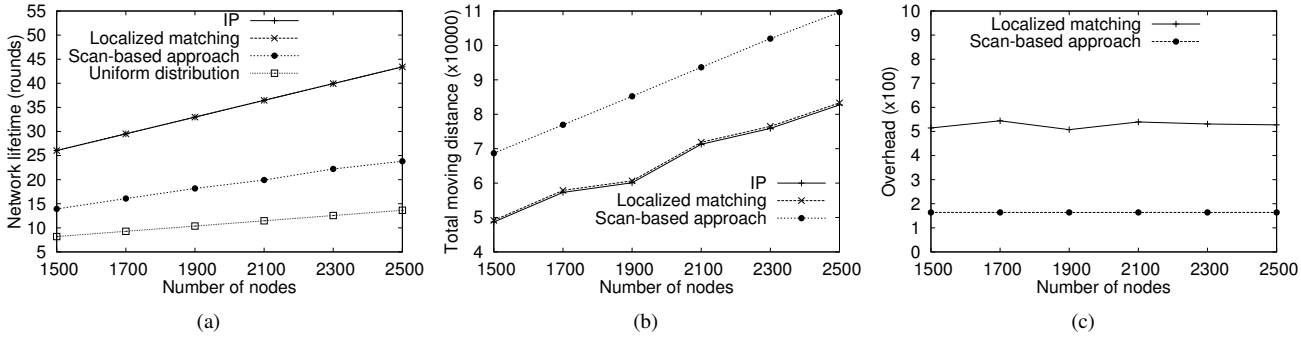

Figure 4. Comparison among IP approach, localized matching method and scan-based approach. (a) Network lifetime comparison, (b) Total moving distance comparison, (c) Overhead comparison.

speeds of TTL (ΔTTL). $\Delta TTL = 1$ 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. $\Delta TTL = 0$ means TTL increases exponentially with base 3, TTL = 1, 3, 9, etc.

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 (the number of rounds the network lasts after the redeployment) and the uniform distribution. IP and localized matching methods have similar performance and they are better than other two. 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.

Network lifetime achieved by the scan-based approach is worse compared to the localized matching for the following reason. In 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 the IP approach, which is the optimal solution. There are two reasons why the total moving distance of the scan-based approach is larger. 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. In our approach, sensors move through regions $t, t-1, t-2, \ldots,$ to region 1, when the optimal way is to move 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. A trade-off between the moving distance and the overhead exists when comparing these two algorithms.

Figure 5 focuses on the localized matching method. In Figure 5a, $\Delta TTL = 1$. 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. 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.

Simulation results in Figures 5c and 5d 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

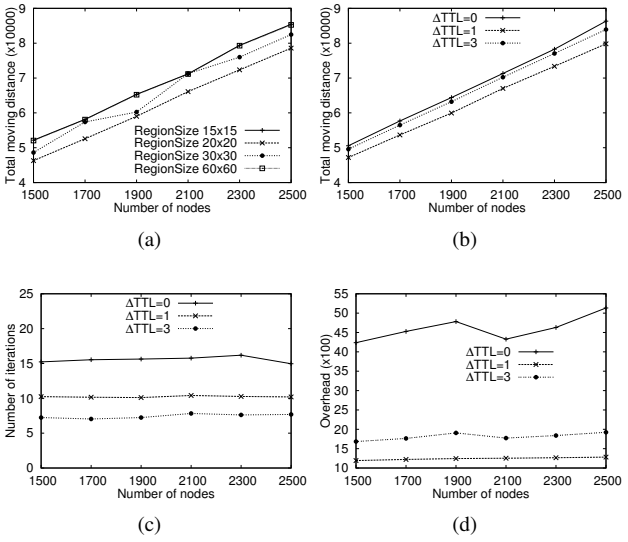

Figure 5. Comparison for the localized matching method.

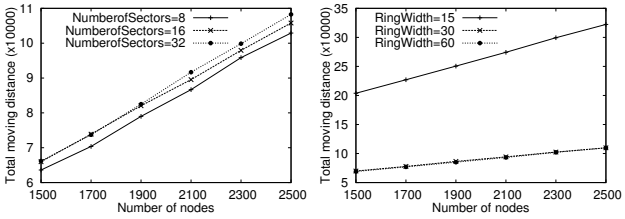

Figure 6. Comparison for the scan-based approach.

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.

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 focuses on the Scan-based approach. In Figure 6a, the ring width is 60 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 moving distance.

5 Conclusions and Future Work

In this paper we focus on sensor redeployment that will prolong network lifetime while minimizing sensor movement. We propose three algorithms to reposition sensors according to desired 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.

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] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

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