Low Cost Region Detection from Distributed Sensor Observations

Chengyang Zhang and Yan Huang
University of North Texas
Denton, TX U.S.A.
Email: {chengyang, huangyan}@unt.edu

***Abstract*—The applications of wireless sensor networks often require region representations of the phenomena that the networks monitor for visualization and interaction with spatial databases. However, there is a fundamental gap between the point-based sensor observation of non-regular sensor deployment and the spatio-temporal phenomena represented by regions. This paper proposes to bridge the gap by providing low-communication-cost approaches to create regions from the distributed sensors. The solution is based on Spatial Grid partition and the task is distributed among a set of sensors called "cell leaders". We formalize the problem of cell-leader selection with the objective of minimizing sensor communication cost. We propose two heuristic algorithms, i.e., Greedy Lattice Search and Subregion Centroid Approximation. We also propose Energy-Aware Local Update model to allow the dynamic re-assignment of cell leaders. The experiments performed on a real sensor observation dataset demonstrated that our heuristic approaches can achieve much better communication cost compared with the baseline methods.**

I. Introduction

The continuing miniaturization of sensors enables a variety of applications that rely on coin- to palm-size computerized sentries. These miniature sensors can identify themselves, locate their positions, and describe their functions as soon as they are plugged into a network [18]. They can also communicate through wireless communication with nearby sensors and transmit data through multi-hop protocols with a gateway, which incorporates long-haul communication capacity. These new developments in wireless self-organized networks enable domain scientists to work with data sets of unprecedented fine spatial and temporal resolution and are revolutionizing many domain sciences.

Many of these sensory devices are geo-referenced wireless sensors. They are deployed for real time monitoring of natural phenomena and alerting of hazard weather conditions. The phenomena that the sensors monitor often need to be represented as regions for visualization and interaction with spatial databases. However, there is a fundamental gap between the point-based sensor observation of non-regular sensor deployment and the spatio-temporal phenomena represented by regions. Due to the resource limitations of miniature sensors, overall energy cost often needs to be minimized for long term monitoring purposes. Because communication is much more expensive than computation, saving communication cost is an effective way to reduce overall energy consumption. Thus, this paper proposes to bridge the gap by providing low-communication-cost approaches to create regions from distributed sensors. The solution is based on a Spatial Grid (*SG*) partition coupled with *spatial interpolation* to handle irregular sensor deployment and *spatial aggregation* to elevate points into regions.

Example 1 (Motivating Scenario: ALERT system) The ALERT system [12] (Automated Local Evaluation in Real Time) is deployed by National Weather Service and other organizations to collect real time hydrologic data (e.g. water level, rainfall) using remote sensors. The data is eventually transmitted to a central server for hydrologic modeling and warning of flood, drought, or hurricane conditions. This system can be enhanced with densely deployed wireless sensor networks to achieve coverage of high spatial and temporal resolutions. Each sensor generates data in the form of *(location, time_stamp, water_level)*. With the assistance of our proposed approaches, before or during a flood or drought, the ALERT system will be able to continuously generates flooding/drought zones, e.g. water level above certain user-given threshold, from distributed sensor readings and send out alerting messages, or identify the damage in real time. The challenge is to allow the sensor network to generate desired regions using interpolation for extended time period without the need to replace batteries.

To the best of our knowledge, our approach is the first attempt to detect and represent phenomena as **regions** using a grid-based interpolation from distributed sensor observations, with the aim of **minimizing communication cost**. Grid-based interpolation is desired because it has been used in many environment monitoring applications and many GIS layers are available in grid format. However in practice, it is very unlikely that the sensors can be uniformly distributed as a grid over a large spatial area due to natural obstacles during deployment. Even we can form such a regular sensor deployment, it is also prone to the failures of certain sensor nodes. As a result, the standard image filtering techniques based on a grid partitioned space cannot be directly applied because of the irregularity of the sensor deployment. The closest work in the sensor network area is the boundary/edge estimation of sensor field [3], [11], [17]. The work aims at identifying sensors located at the edges of the phenomena to handle the irregular sensor placement. The methods include statistical and classifier-based approaches. Our work is different in that we use spatial interpolation to handle the irregularity

978-0-7695-4048-1/10 $26.00 © 2010 IEEE
DOI 10.1109/MDM.2010.49

of sensor field. We focus on minimizing communication cost and propose two heuristic algorithms to distribute the task of energy-efficient generation of the regions among a set of sensors called *cell leaders*.

This paper brings together the following contributions:

1) We propose low-communication-cost approaches to detect regions from distributed point sensor observations. The approaches handle irregular sensor field by *spatial interpolation* and elevate points into regions using Spatial Grid (*SG*) partition.
2) We formalize the problem of cell-leader selection with the objective of minimizing sensor communication cost (to reduce overall energy use). We show that a global optimal solution is computationally prohibitive for high resolution grids. Then we propose two heuristic algorithms, i.e., Greedy Lattice Search (*GLS*) and Subregion Centroid Approximation(*SCA*). We further propose a dynamic cell-leader re-assignment approach to alleviate the problem of unbalanced energy consumption of cell-leaders and other sensors.
3) We compare the proposed algorithms experimentally with the schemes based on random selection and nearest sensors respectively. The experiments performed on a real sensor observation dataset demonstrated that our heuristic approaches are scalable to high grid resolution. The results also show that the GLS and the SCA approaches can achieve much better communication cost compared with the baseline methods.

The rest of the paper is organized as follows: Models and preliminaries are presented in section II. Section III proposes two heuristic algorithms for cell-leader selection. Then experiment results are presented in section IV. Related work are reviewed in section V. Finally, section VI concludes the paper and discusses future work.

II. System Model and Preliminaries

In this section, we first present the system model and problem definition. Then we give notations and assumptions used in the paper. Last we discuss the influence of several important parameters.

A. System Model

Our system consists of a wireless sensor network and a centralized server. The sensor network is deployed to monitor a natural phenomenon (e.g. hot zones) represented by ridges (projected as regions in red color) as shown in Figure 1. The location of each sensor (dot in the figure) is given. The network is connected and the sensors can communicate with the centralized server through a sink node. Information of regions is collected locally and sent to the centralized server where such information is summarized and presented to the end user.

On the centralized server, we use a Spatial Grid (*SG*) to partition and interpolate a 2D space. Compared with other partition methods, e.g. Voronoi Cells, SG is less affected by the irregularity of the sensor deployment and allows for different

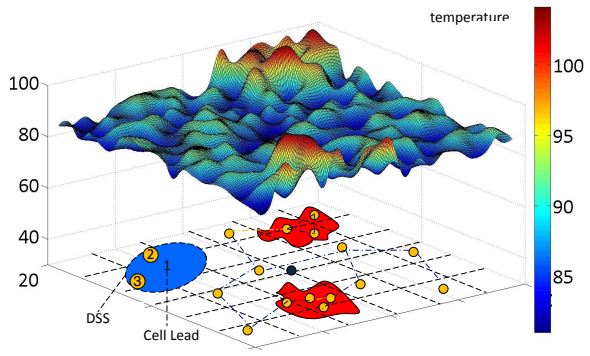


Fig. 1. System Illustration

interpolation functions to be applied. The detected regions are represented using grid cells in a way that resembles the raster pixels. The resolution of the grid is a user-defined parameter.

Each grid cell is associated with a set of sensors within a certain distance of that cell. We call these sensors the cell's **Dominating Sensor Set** (DSS). For example, the shaded cell (blue cell in color print) in Figure 1 has three sensors (1,2 and 3) around it for a given interpolation distance. These three sensors constitute the shaded cell's DSS. A cell's value can be interpolated using all the sensors in that cell's DSS. The sensors not in a cell's DSS do not affect that cell's interpolation value because they are too far away. With given interpolation radius and grid resolution, each sensor can determine the set of cells associated with it.

For each cell we select one sensor from the network as the **Cell Leader** (CL). CL is responsible for collecting all the information for that cell from the cell's DSS. For example, the shaded cell in Figure 1 may select sensor 1 in its center as the cell leader. Sensor 1 will collect observation values from sensors 2 and 3 and together with its own reading, interpolate the shaded cell's value using a weighted function. With the interpolated value, sensor 1 can determine whether the shaded cell belongs to the monitored phenomenon based on a user-given predicate (e.g., "temperature higher than 100F").

Please note that CL itself is not necessarily in the cell's DSS. Furthermore, a CL may be responsible for multiple cells. The CLs aggregate the cells that satisfy the predicate into micro-regions. Then the micro-regions are sent to the centralized server through the sink node (black dot in Figure 1). The server will merge all the micro-regions and provide final region results to the end user.

For each time snapshot, to generate the desired regions, communication cost is incurred when CLs collect each cell's DSS information and when CLs send the sub-region information back to the server. Strictly speaking, communication cost is affected by many factors including channel interference etc. However, in general it is largely dependent on the hop distances of all the messages sent in order for the server to generate the regions. In this paper we define *message distance(md)* between two sensors as the number of hops

required for a message to be transmitted from one sensor to the other. The overall communication cost is measured by the total *md*s of all the messages sent. To achieve low communication cost, CLs need to be carefully selected in order to minimize the overall message distances. Ideally the CLs should also be locally determined and dynamically updated according to energy constraints and network configurations.

B. Problem Definition

In a 2D space monitored by a fixed sensor network with a single sink node, given the grid resolution, interpolation radius, and the phenomenon predicate (e.g. temperature above 100F), find the set of cell leaders to interpolate and aggregate cells into regions, with the objective of minimizing communication cost.

A phenomenon/region is defined as a set of connected cells with interpolation values that satisfy the user-given predicate.

C. Notations and Assumptions

The notations used in the paper are illustrated in Figure 2 and summarized in Table I. The integer number over each link in Figure 2 denotes the *message distance* of the link. The real numbers over the links are the Euclidean distances.

In this paper, we assume connected sensor networks. Therefore each sensor can always communicate with any other sensor in the network through multi-hop communications. If there are multiple disjoint networks, the approaches in the paper can be applied to each network independently. For simplicity, we assume that the interpolation radius r_i is the same for all the cells. However, this condition is not required for our approaches to work. We also assume a single sink node in this paper.

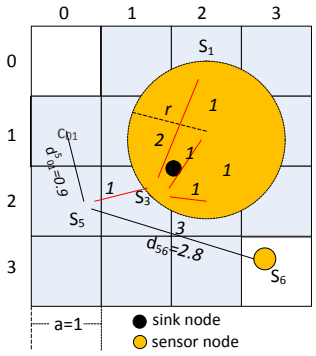


Fig. 2. Illustrative Example

We partition the space using a Spatial Grid (*SG*). For convenience, we use simple average function to interpolate each cell's value. However, SG allows various weight function to perform arbitrary linear interpolation. The details of interpolation are out of the scope of this paper.

D. Influence of Parameters

Now we briefly discuss the influence of some important parameters in the system.

- Grid resolution: grid resolution is a user-defined parameter that allows the trade off between the quality of desired region and the communication cost. When grid resolution increases, a sensor will need to interpolate and aggregate information from more cells, resulting higher communication cost. When the grid resolution is fixed, our system can always find the exact region with grid cell as the minimum unit.
- Number of sensors: as the number of sensors increases, the resultant region(s) are better representations of the phenomena because of higher sampling accuracy. However, this comes with the price of increasing communication cost.
- Interpolation radius: the interpolation radius determines how far away a sensor can be while still having impact on the interpolation values on a location. It is determined by the characteristics of monitored phenomena. When interpolation radius increases, the overlapping of each cell's DSS also increases, resulting higher message cost.

III. Cell Leader Selection Schemes

In this section, we will use the notations and assumptions provided in the previous section to design schemes for assigning each cell a cell leader (CL). Our objective is to minimize in-network communication cost incurred when CLs collect information from other sensors and when CLs send summary information to the sink node. As discussed earlier, we focus on minimizing overall message distances.

Example 2 (Running Example: Selection of CLs) The illustrative example in Figure 2 shows a sensor network consisting of 6 sensors labeled as s_1 to s_6 for a spatial partition of $G[4\times 4]$. We now examine how the CLs should be selected in this simple case. For simplicity of illustration, we assume that $r_i = a = 1$ (a is the length of the cell edge), therefore a cell's DSS contains all the sensors within 1 distance of that cell's center location. For example, cell c_{21}'s DSS includes sensor s_1 to s_4 that are inside the circle (orange circle in color print). The message distance of each link is also marked in the figure. And we ignore the cost at summary stage at this time.

We find that $DSS(c_{00}) = \phi$, and therefore no cell leader is required for cell c_{00}. For cell c_{21}, if we choose s_2 as the cell lead, the message cost will be $1+1+1=3$. If we choose s_3 as the cell leader instead, the cost will be $1+1+2=4$. If we choose s_6 which is out of c_{21}'s interpolation range, the cost is even higher. Similarly for any other cell, we can always scan all the sensors to find a cell leader that minimizes the sum of message distance to all the sensors in that cell's DSS. We denote such a selection **local optimal** for that cell.

Note however, a sensor may be CLs of multiple cells (even when the sensor is not in a cell's DSS since we assume the sensor network is connected). Therefore many duplicate messages may occur if we only choose local optimal solution for each individual cell. For example, the local optimal solution for c_{21} is s_2 with the cost of 3; and the local optimal solution for cell c_{12} is s_3 with the cost of 3. If we choose s_2 for c_{21}

TABLE I
LIST OF NOTATIONS

Notation	Explanation
$G[n \times n]$	Grid partition of the space into $n \times n$ cells, e.g. Figure 2 shows $G[4 \times 4]$.
a	The length of the edge of a cell, e.g., in Figure 2, $a = 1$.
c_{ij}, $0 \leq i, j \leq n$	A cell with the top-left corner as origin, e.g., the dark (blue in color print) cell in Figure 2 is c_{01}.
s_k, $0 \leq k \leq m$	The kth sensor node among m sensors, e.g., 6 sensors in Figure 2 are denoted as s_1 to s_6.
ϵ_k, $0 \leq k \leq m$	The percentage of the energy remaining in the kth sensor node. When a sensor's energy drops below a threshold, it must enter sleep mode and wake up again when the energy is recovered.
$f(\epsilon_k)$, $0 \leq k \leq m$	The energy function used to determine whether a sensor needs to enter sleep mode.
sn	The sink node located at the center of the space.
$d_{ij} = \|s_i - s_j\|$	The Euclidean distance between sensor s_i and s_j, e.g, $d_{56} = 2.8$.
$d_{ij}^k = \|s_k - c_{ij}\|$	Distance between sensor s_k and cell c_{ij}'s center location. E.g., $d_{01}^5 = 0.9$.
r_c	The communication range of each sensor
$md_{i,j} = d_{ij}/r_c$	The *message distance* defined as the number of intermediate messages needed for sensor i to send a message to sensor j. E.g., $md_{1,2} = 1$, $md_{5,6} = 3$. The $md_{i,j}$ defined here is an approximation of the real message distance. The real number of hops between sensors i and j depends on the network configuration.
r_i	The interpolation radius. Sensors more than r_i away from the cell does not have effect on the cell's value. r_i is different from r_c and depends on the properties of the phenomena.
$DSS(c_{ij}) = \bigcup\{s_k \| d_{ij}^k \leq r_i\}$	Dominating Sensor Set (DSS). DSS is a set of sensors within r_i of the center of the cell c_{ij}. $DSS(c_{ij})$ contains all the sensors that can interpolate the value at c_{ij}. E.g., $DSS(c_{21}) = \{s_1, s_2, s_3, s_4\}$ contains 4 sensors in the shaded circle (orange in color) in Figure 2.
l_{ij}	The cell leader of cell c_{ij}. Cell leader l_{ij} is a sensor responsible for interpolating c_{ij}'s value by collecting information from $DSS(c_{ij})$. E.g., choose s_2 may be the cell leader for c_{32} in Figure 2.

and s_3 for c_{12}, the total cost is $3 + 3 = 6$. If we choose s_3 as the common cell leader for both c_{21} and c_{12}, the actual cost is only 5 after eliminating duplicate messages. This indicates that the simple combination of the local optimal CL selections is not necessarily the global optimal solution in general.

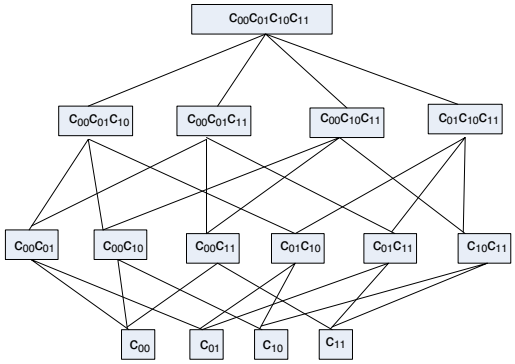
A. Finding Global Optimal Solution

To find the global optimal solution for more than one cell, a naive approach is to find all possible combinations of CLs for all cells and pick the best. Assuming the total number of sensors is m, because each cell can choose one of the m sensors as its CL, there will be $(m)^{n^2}$ possible combinations for the partition $G[n \times n]$. For each combination, we compute the cost based on each cell's DSS set. In the worst case, such an approach has the computational cost of $O((m)^{n^2} \cdot m^2)$. This is computationally prohibitive even when grid resolution is not very high.

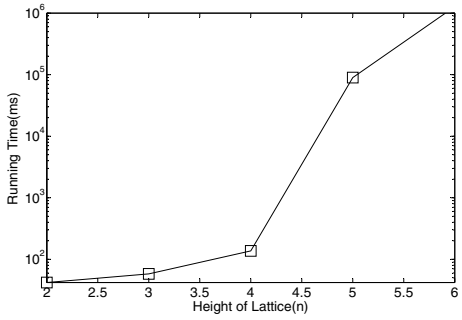
Consider the case of two cells $c_{i_1 j_1}$ and $c_{i_2 j_2}$. An optimal solution (in the above sense) can be achieved by evaluating two alternatives: using two local optimal CLs, or use one common CL for two cells. The former has less message distance for each individual cell but may incur duplicate messages, while the latter has no duplicate messages but may have higher cost for some cell. Note that we will also need to consider the cost from the cell leader to the sink node.

The case can be extended to more cells by evaluating more combinations of optimal CLs, which is facilitated using a cell lattice illustrated in Figure 3(a).

As Figure 3(a) illustrates, the lattice D contains n^2 layers ($n = 2$ in the figure), and we denote the lth layer ($l = 1$ to n^2) as D_l. D_l has $C_{n^2}^l$ nodes. Each nodes is a combination of l cells. The optimal solution for this node can be found by evaluating the cost of using a single cell leader for all the cells in this node, and the cost of using multiple cell leaders computed from the results of lower layers. We can apply a dynamic programming approach to scan the lattice in a bottom-up fashion, and the global optimal solution is found when we reach the top of the lattice.



(a) Cell Lattice for a 2x2 Grid



(b) Running time w.r.t n

Fig. 3. Illustration of Cell Lattice

The detailed algorithm to find global optimal solution is shown in Algorithm 1. Note that the *for* loop at Line 15 evaluates all the subsets of the node. In actual implemen-

Algorithm 1 Finding Global Optimal Solution

1: Initialize the Lattice D for $G[n \times n]$
2: Let D_l ($l = 1$ to n^2) denote the lth layer of D, $D_l(i)$ ($i = 1$ to $C^l_{n^2}$) denote the ith node in D_l (left to right), and $CAN(D_l(i))$ denote the candidate CL selections for $D_l(i)$
3: //Layer 1
4: **for** $i = 1$ to n^2 **do**
5: Find s_k so that $md_{k,sn}+\sum_{x\in DSS(D_1(i))} md_{k,x}$ is minimum among all the sensors
6: Optimal solution for $D_i(i)$ is s_k and its associated cost.
7: **end for**
8: //layer 2 and above
9: **for** $l = 2$ to n^2 **do**
10: **for** $i = 1$ to $C^l_{n^2}$ **do**
11: //single cell lead
12: Find s_k so that $md_{k,sn} + \sum_{x\in DSS(D_l(i))} md_{k,x}$ is minimum among all the sensors
13: $CAN(D_l(i)).add(s_k)$. Store $CAN(D_l(i))$ and its associated cost set.
14: //multiple cell leaders
15: **for** any $D_{l'}(j)$ so that $l' < l$ and $D_{l'}(j) \subset D_l(i)$ **do**
16: Find $D_l(i) - D_{l'}(j)$ from layer $D_{l-l'}$
17: Combine the optimal solutions for $D_l(i) - D_{l'}(j)$ and $D_{l'}(j)$ as a solution for $D_l(i)$, and add this solution to $CAN(D_l(i))$
18: **end for**
19: Optimal solution for $D_l(i)$ is the MIN COST over $CAN(D_l(i))$
20: **end for**
21: **end for**

tation, we may mark the visited subset to avoid duplicate computation. However, the computation cost is still very high even with the lattice. When the number of cells is large, the solution is also computationally prohibitive. Figure 3(b) shows an example implementation executed on a server with two with two 2.33 GHz Intel Core 2 Duo CPU and 3GByte memory. When $n = 6$, i.e. the space is partitioned using 36 cells, the computation time is already almost unacceptable. Therefore heuristic algorithms are required that are scalable to high grid resolutions while still achieving comparably low communication cost.

B. Proposed Heuristic Solutions

The global optimal solution is computationally prohibitive. A good heuristic should be able to prune the large search space to improve performance while still achieving relatively low message cost. Another important issue is that the cell leaders are hot-spots and their energy consumption is much higher than that of the other sensors. Ideally a method should be able to dynamically rotate the cell leaders based on their energy conditions. We will provide two heuristics to find initial solution and then propose a dynamic localized rotation scheme to balance energy consumption in the next subsection.

1) Heuristic 1: Greedy Lattice Search (GLS): Assuming initially all the sensors are running with full energy, the cost function of a sensor s_k as the cell leader of a cell c_{ij} can be calculated as:

$$cost^{ij}_k = md_{k,sn} + \sum_{\forall s_x \in DSS(c_{ij})} md_{k,x} \tag{1}$$

In words, the cost of s_k as the cell leader of cell c_{ij} is the total message distance between s_k and the sink node, plus the sum of of message distances between s_k and any sensor in c_{ij}'s DSS. By definition, a cell c_{ij}'s local optimal solution may be any sensor in the space. However, we have following simple observation:

Observation 1: The local optimal cell leader selection for any cell tends to be inside that cell's DSS set.

Based on observation 1, we may approximate a cell c_{ij}'s optimal solution s^{ij}_{opt} by searching sensors within its DSS. We have:

$$s^{ij}_{opt} = \underset{\forall s_k \in DSS(c_{ij})}{\arg\min} \; cost^{ij}_k \tag{2}$$

In words, the local optimal cell leader of cell c_{ij} denoted as s^{ij}_{opt} is the sensor in c_{ij}'s DSS set that minimizes the cost of interpolating the value of cell c_{ij} and transmitting the value to the sink if above threshold. With Equation 1 and 2, the heuristic solution for bottom layer of the lattice (illustrated in Figure 3(a)) can be determined.

Next we go through the lattice in a bottom-up fashion and try to greedily combine the CLs of some cells to eliminate duplicate messages. At each step, we merge the cell leaders of two nodes in the lattice if the overall message cost can be reduced. If there are multiple choices, we choose the one that can best reduce the cost. The two nodes to be merged may cross different layers. The merged nodes and its children in the lattice will be marked as "merged" to prune the search space. A new merged node will be generated. Only unmarked nodes will participate in the subsequent merges. The process stops when we reach the top of the lattice or the overall message distance cannot be further reduced.

To facilitate the search process, we maintain a set $\mathcal{S}$ that initially contains all the cells together with their optimal cell leaders. When we merge two nodes in the lattice, the nodes and their cell leaders are removed from $\mathcal{S}$, while the upper layer node formed through merge operation will be added to $\mathcal{S}$ together with the new cell leader. The optimal solution is found when elements in $\mathcal{S}$ cannot be further combined (either there is only one element in $\mathcal{S}$, or any combination will not reduce the overall cost).

In order to evaluate the cost during the merge process, we extend Equation 1 to describe the cost of sensor s_k as the leader of a node e in the lattice. The node e is in fact a set of cells.

$$cost^e_k = md_{k,sn} + \sum_{\forall s_x \in \cup\{DSS(c_{ij})|\forall c_{ij}\in e\}} md_{k,x} \tag{3}$$

Similarly, Equation 2 becomes:

$$s^e_{opt} = \underset{\forall s_k \in \cup\{DSS(c_{ij})|\forall c_{ij}\in e\}}{\arg\min} \; cost^e_k \tag{4}$$

Thus the decision of merge operation is always made based on the following condition:

Merge Condition: Two nodes e_1 and e_2 in the lattice may merge into one upper layer node $e_3 = e_1 \cup e_2$ if:

$$e_1 \cap e_2 = \phi$$
$$e_1 e_2 = \arg\max_{\forall e_i e_j \in \mathcal{S}} (cost_{k_1}^{e_1} + cost_{k_2}^{e_2} - cost_{k_3}^{e_3}) \quad (5)$$
$$cost_{k_1}^{e_1} + cost_{k_2}^{e_2} > cost_{k_3}^{e_3}$$

where $s_{k_1} = s_{opt}^{e_1}, s_{k_2} = s_{opt}^{e_2}, s_{k_3} = s_{opt}^{e_3}$.

Note that in Equation 5, s_{k_3} is chosen from the set of all sensors in the DSS of node e_3.

In summary, the GLS algorithm proceeds as follows:

1) First the Lattice shown in Figure 3 (a) is initialized. For the bottom layer of the lattice, we scan each cell's DSS to find an initial cell leader that minimizes the cost function within the DSS.
2) In the next stage the cell lattice is traversed in a bottom-up fashion. Based on equation 5, we choose to merge the nodes that will best reduce the overall message distance.
3) The process stops when the cost cannot be further reduced or we reach the top of the lattice.

The pseudo-code of GLS algorithm is given in Algorithm 2.

Algorithm 2 Greedy Lattice Search(GLS)

```
Initialize Lattice D for G[n × n] and set S
for i = 1 to n do
  for j = 1 to n do
    Find s_opt^ij based on Equation 1 and 2
    Add c_ij to S. Store s_opt^ij and the optimal cost associated with c_ij
  end for
end for
while S.size> 1 do
  if any pair of nodes e1 and e2 in S satisfies equation 5 then
    remove e1 and e2 from S. Clear cost associated with them as well.
    add e3 = e1 ∪ e2 to S. Store s_opt^e3 and the optimal cost associated
    with e3
  else
    //The cost cannot be further reduced
    EXIT SYSTEM
  end if
end while
```

2) Heuristic 2: Subregion Centroid Approximation (SCA): Intuitively, if we divide the space into subregions and use a leader to interpolate a subregion instead of a cell, the cost of communication may be reduced. This is based on the inherent property of spatial auto correlation in spatial phenomena. We term such a solution as Subregion Centroid Approximation (SCA). The basic idea is to divide the space into non-overlapping subregions. Then we use the sensor closest to the center/centroid of the subregion as the cell leader for all the cells inside this subregion. We assume that subregions are relatively large and there are many sensors so that a sensor exists close to the center of a subregion. We further assume in the calculation that this sensor is located precisely at the center of the subregion. We have the following observation:

Observation 2: The subregion should be neither too large nor too small. Large subregion would have longer average message distances, while small region may introduces more duplicate messages.

Thus, the challenge is to algebraically derive the proper size of the subregion which will be the focus of this section.

We partition $G[n \times n]$ with non-overlapping subregions of length $a' = \kappa a$. For simplicity, we assume κ and n/κ are both integers. (The approach can be easily extended to non-integers). Therefore the number of subregion is n^2/κ^2, and each subregion contains $\kappa \times \kappa$ cells. The number of cell leader is also n^2/κ^2 since each subregion has exactly one CL at the center. Figure 4 illustrates the $G[9 \times 9]$ partitioned by 3×3 subregions. Each subregion has 3×3 cells, i.e., $n = 9$, $\kappa = 3$.

Now we estimate the overall message cost as follows. We first compute the cost from the cell leaders to the sink node.

Theorem 3.1: In SCA approach, the overall cost from the cell leaders to the sink node is $\frac{n(n^2-\kappa^2)}{2\kappa^2 r_c} \cdot a$ if n/κ is an odd number, and $\frac{n^3}{\kappa^2 r_c} a$ if n/κ is an even number.

Proof: Assuming n/κ is an odd number, the cost from the center subregion's cell leader to the sink node is 0. We first consider one row of the partition. The horizontal message distances between all cell leaders at this row and the sink can be computed as $2\sum_{i=1}^{\frac{n/\kappa-1}{2}} i \cdot a'/r_c$ since the partition is symmetric. The vertical distances are the same for them. Since the partition has n/κ rows, the overall distances can be computed as $4 \cdot n/\kappa \cdot \sum_{i=1}^{\frac{n/\kappa-1}{2}} i \cdot a'/r_c = \frac{n(n^2-\kappa^2)}{2\kappa^3 r_c} \cdot a' = \frac{n(n^2-\kappa^2)}{2\kappa^2 r_c} \cdot a$. Similarly, when n/κ is even, the cost can be computed as $\frac{n^3}{\kappa^2 r_c} a$. ■

Next we estimate the message distances from the cell leader to all the sensors in the subregion's DSS.

Theorem 3.2: Given interpolation range r_i, transmission range r_c,cell length a, subregion length $a' = \kappa a$, number of sensors m, the overall cost from each cell leader to all the sensors in the subregion's DSS is approximately $\frac{4r-2a+2\kappa a}{r_c} m$.

Proof: As Figure 4 shows, the union of interpolation range for all the cells in the subregion may be slightly bigger than the subregion itself because the border cells' interpolation range may overlap with other subregions. We denote this range using a square (red square in the color print of Figure 4). And it is easy to see the edge length of the square is $2r - a + a'$. It can be proved that the average distance from the cell leader to all the points in the interpolation range is $\frac{r-a/2+a'/2}{r_c}$. Since there are a total of m sensors in the space, it can be seen from the figure that the maximum number of duplicate messages is 4 times considering all the neighbors of the subregion. Therefore the total distance is approximately $\frac{r-a/2+a'/2}{r_c} \cdot 4m = \frac{4r-2a+2\kappa a}{r_c} m$. ■

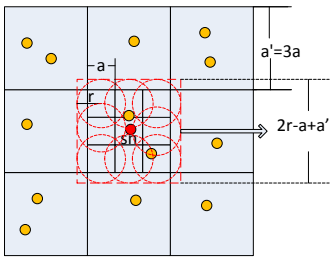


Fig. 4. Illustration of SCA

Combining Theorem 3.1 and 3.2, we have:

Theorem 3.3: The SCA approach's cost is minimized when $\kappa = \lceil \sqrt[3]{\frac{n^3}{m}} \rceil$ if n/κ is even, and $\kappa = \lceil \sqrt[3]{\frac{n^3}{2m}} \rceil$ if n/κ is odd.

Proof: Let's consider when n/κ is an even number. From Theorem 3.1 and 3.2, the total cost is a function of κ, i.e., $g(\kappa) = \frac{4r-2a+2\kappa a}{r_c} m + \frac{n^3}{\kappa^2 r_c} a$. To minimize $g(\kappa)$, we can prove that $g''(\kappa) > 0$. Therefore we set $g'(\kappa) = 0$. The solution is $\kappa = \lceil \sqrt[3]{\frac{n^3}{m}} \rceil$. When n/κ is an odd number we can get the solution similarly. ■

C. Energy-Aware Local Update (EALU)

In our heuristic approaches, we did not consider the energy conditions of each sensor. However, in practice the cell leaders are hot-spots and consume more energy than other sensors. In this subsection, we propose a localized CL rotation scheme based on peer search. Given sensor s_k, we find all the cells with s_k as the CL. Then we get the union of the DSS set of these cells, and compute s_k's energy-aware cost as:

$$cost_k = \frac{md_{k,sn} + \sum_{\forall s_x \in \{\cup DSS(c_{ij}) | l_{ij} = s_k\}} md_{k,x}}{f(\epsilon_k)} \tag{6}$$

In Equation 6, $f(\epsilon_k)$ is sensor s_k's energy function. For example, we may use the following function:

$$f(\epsilon_k) = \begin{cases} 1 & \text{if } \epsilon_k > \theta_s \\ \epsilon_k & \text{if } \theta_h \le \epsilon_k < \theta_s \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

In Equation 7, θ_s and θ_h are soft threshold and hard threshold respectively. When a sensor's energy drops below a soft threshold, the energy status of the node starts to be taken into consideration in the cost estimation. When a sensor's energy drops below a hard threshold, it will enter the sleep mode immediately and the cost is therefore infinity.

Case 1: When a sensor s_k detects its own energy condition drops below the soft threshold θ_s, it will probe its neighborhood to find a list of sensors $Nbr(s_k)$. $Nbr(s_k)$ contains all the sensors in s_k's neighborhood not just the cell leaders. Then it will check each sensor in the $Nbr(s_k)$, and try to delegate its CL task to a sensor s_l in the $Nbr(s_k)$ to best reduce the overall cost. When s_k delegate CL to s_l, the reduction of cost $\Delta cost_{kl}$ is:

$$\Delta cost_{kl} = cost_k + cost_l - \frac{md_{l,sn} + \sum_{\forall s_x \in \{\cup DSS(c_{ij}) | l_{ij} = s_k \vee s_l\}} md_{l,x}}{f(\epsilon_l)} \tag{8}$$

The best s_l to choose must satisfy

$$s_l = \underset{\forall s_x \in Nbr(s_k)}{\arg\max} \ \Delta cost_{kx} \quad \text{where } \Delta cost_{kl} > 0 \tag{9}$$

If Equation 9 is not satisfied for any sensor in $Nbr(s_k)$, then s_k will wait for a while and try the above process again.

Case 2: When a sensor s_k's energy is below the hard threshold θ_h, the first condition in Equation 9 is trivially satisfied as $cost_k = \infty$. We denote

$$\Delta' cost_{kl} = cost_l - \frac{md_{l,sn} + \sum_{\forall s_x \in \{\cup DSS(c_{ij}) | l_{ij} = s_k \vee s_l\}} md_{l,x}}{f(\epsilon_l)} \tag{10}$$

Then s_k will delegate CL to a sensor s_l in $Nbr(s_k)$ so that

$$s_l = \underset{\forall s_x \in Nbr(s_k)}{\arg\max} \ \Delta' cost_{kx} \tag{11}$$

The pseudo-code of EALU algorithm is detailed in Algorithm 3.

Algorithm 3 Energy-Aware Greedy Localized Update(EALU)

```
//For any sensor s_k
if f(ε_k) < θ_s then
  while f(ε_k) ≥ θ_h do
    Probe s_k's neighborhood to get a list of sensors Nbr(s_k)
    Compute s_l based on Equation 9 for Nbr(s_k)
    if Equation 9 satisfies then
      delegate s_k's task to s_l
      EXIT WHILE LOOP
    else
      //Wait a while and retry
    end if
  end while
end if
if f(ε_k) < θ_h then
  Probe s_k's neighborhood to get a list of sensors Nbr(s_k)
  Compute s_l based on Equation 11 for Nbr(s_k)
  delegate s_k's task to s_l
end if
```

In EALU algorithm, a sensor tends to detect more neighbors when communication ranges increases. Therefore the delegation of cell leader task may take longer, but more likely to succeed. The execution of EALU algorithm is largely dependent on the selection of appropriate energy thresholds. Higher θ_s implies frequent local update but more reliable leaders. Higher θ_h implies more sensors entering sleep mode.

In summary, the GLS algorithm always greedily merges the cells in the lattice to approximate the global optimal solution. The SCA approach tries to adjust subregion size to reduce cost by trading off between average distance and duplicated messages. Finally the EALU algorithm uses the localized neighbor search to handle energy problem.

IV. Experiments

In this section, we will present the experiment results to demonstrate the effectiveness of our heuristic algorithms. We compare our proposed algorithms with two baseline methods: the scheme based on random selection and the scheme based on nearest sensors. In the first baseline scheme, we choose a CL for each cell by randomly picking a sensor from the DSS set of that cell. In the second baseline method, we assign each cell a CL such that the CL sensor is the one that is closest to the cell's center. The two baseline methods are denoted as *RAN* and *NN* respectively. We did not choose a global optimal solution as the baseline because as Figure 3(a) shows, such solution for large number of cells is un-obtainable.

Note that the goal of this paper is to design algorithms to allow low cost region construction. Regardless of the effectiveness of the CL selection, the quality of the returned regions

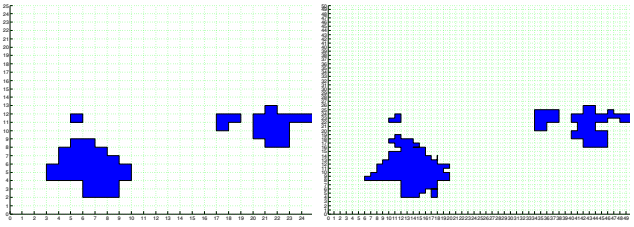
is guaranteed and only dependent on the user-defined grid resolution. Thus, only energy consumptions of the proposed algorithms needs to be compared in our experiments although we show the same regions detected by all algorithms in one of our experiments.

We use a modified version of soil moisture data archive provided by TEO project [16] for experiments. The archive contains soil moisture data within an area in Greenbelt Corridor Park near Denton, TX. We generate new data points to cover a larger square area of 1 $mile\times$ 1 $mile$. The sensors are randomly generated in this area. The number of sensors range from 100 to 1000. We intentionally designed several "high moisture zones" serving as the regions need to be detected. If a sensor is deployed inside such a zone, the observation value of the sensor is higher than a given threshold (e.g., 30% relative moisture percentage). Otherwise the value is lower than the threshold. By default, we choose simple average interpolation for all the cells.

In the default settings, $r_i = 0.2\,mile, m = 250, n = 25$ $(i.e., a = 0.04\,mile), r_c = 0.06\,mile$. To simulate the energy consumption of sensors in EALU algorithm, we use a widely accepted energy cost model and adapt the model to message distances. In this model,the energy cost of sending a message from sensor s_k to sensor s_l is $\alpha_{s,l} = a + b \cdot (md_{k,l} \cdot r_c)^{\gamma}$, where we set $a = 50nJ/bit, b = 100pJ/bit/m^2, \gamma = 2$ as suggested in [2]. The full energy ϵ_f is selected as $10000J$. We assume that the messages are relayed as soon as they are received. The default message size is set to 2Kb.

A. Effect of Grid Resolution

We first measure the effect of grid resolution (n) on the quality of generated regions and communication cost. Note that n is a user specified parameter. When n is fixed, our heuristics can always produce regions with the same quality as the global optimal solution. We set n between 10 and 50, resulting in grids of sizes from 10×10 to 50×50 cells. As



(a) n=25 (b) n=50

Fig. 5. Effect of Grid Resolution (n) on Quality of the Region

demonstrated by Figure 5, when n increases from 25 to 50, many details of region boundary emerged. In some cases of our experiments, one region may even split into several regions as n increases because some of the cells are removed from the original region when they are found not eligible under finer granularity.

Figure 6 demonstrates the effect of grid resolution on communication cost. As shown in the figure, our GLS algorithm outperforms the baseline algorithms significantly. The

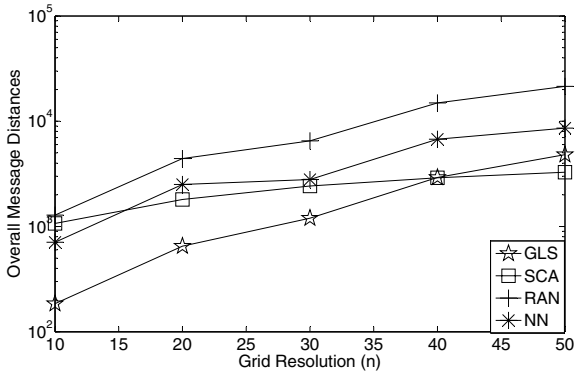


Fig. 6. Effect of Grid Resolution n on Communication Cost ($r_i = 0.2\,mile, m = 250, n = 10 - 50$)

reduce of message cost is about 4 times compared with NN scheme, and nearly 7 times compared with RAN scheme. The SCA algorithm also performs better than baseline methods especially when n is large. This indicates that our heuristics are effective in practice. We observe that the performance of SCA approach tends to be less affected by the change of n, due to the fact that it adjusts the subregion size based on the value of n. In general, when n is small, we would prefer using the GLS method because it performs very well. When n is large, SCA exhibits the better performance and should be chosen.

B. Effect of Number of Sensors

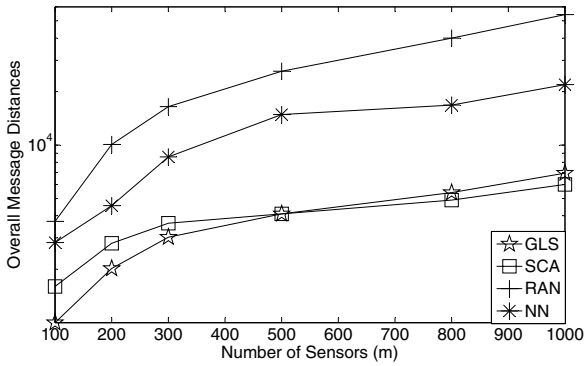


Fig. 7. Effect of Number of Sensors(m) on Communication Cost ($r_i = 0.2\,mile, m = 100 - 1000, n = 25$)

Next we evaluate the effect of number of sensors on the communication cost. From Figure 7 it can be seen that the communication costs of the two proposed algorithms are 2 to 4 times better than NN scheme, and 3 to 10 times better than the RAN scheme. We see that RAN scheme is less scalable to the change of number of sensors as the duplicate messages tend to increase with random selection from increasing number of sensors. The two heuristic algorithms perform similarly with SCA exhibiting slightly better scalability with respect to the number of sensors. It is because SCA will adjust the size of the subregion to make sure each subregion does not contain too

many sensors. The GLS's greedy and non-reversible merging approach becomes less effective when m is relatively large.

C. Effect of Interpolation Radius

The interpolation radius r_i's effect on the communication cost is shown in Figure 8. We observe that the GLS algorithm and baseline methods exhibit similar effects as above. GLS still outperforms the baseline algorithms significantly. But we notice that SCA seems cannot handle the change of r_i as well as the GLS approach. As r_i increases, the advantage of GLS is more prominent. This is because the size of the subregion in SCA is not adjusted according to r_i. A larger r_i would inevitably increase the duplicate messages. By contrast, GLS still relies on the merge of cell leaders to reduce duplicate messages. Therefore it exhibits robust performance to the variation of r_i.

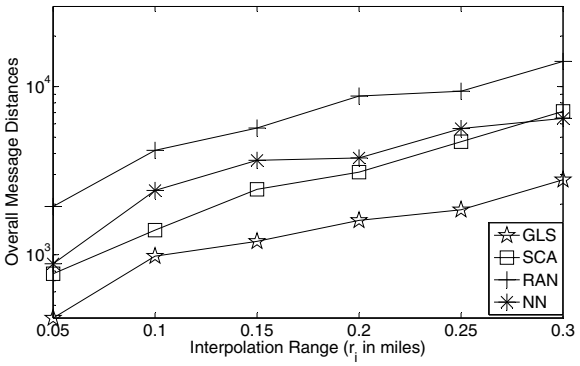


Fig. 8. Effect of Interpolation Range (r_i)on communication cost ($r_i = 0.05\,mile - 0.3\,mile, m = 250, n = 25$)

D. Effect of Energy Threshold

Finally we evaluate the effectiveness of our Energy-Aware Localized Update Algorithm in maintaining the health status of the whole sensor network. The initial cell leaders are selected by GLS algorithm. We monitor the change of energy for an interval of 10 days with sampling rate of 30 seconds. We assume that when a sensor enters sleep mode, it will take 2 hours (through solar panels for example) to recharge and wake up. Then we measure the average percentage of active nodes during the time interval, and compare the value with/without EALU enabled w.r.t the change of soft threshold θ_s. For each θ_s we run the experiments 5 times with random sensor positions and the results are averaged. The value of θ_s ranges from 50% of full energy to 10% of full energy. And the hard threshold θ_h is chosen to be 5% of full energy. We use default values for all other parameters. As Figure 9 shows, EALU algorithm significantly improve the overall number of active nodes even additional communication overhead is added to search neighbors in cell leader delegation process.

V. Related Work

Our work is focused on creating region representations for the phenomena monitored by distributed sensors, with the aim of minimizing communication cost. Closely related work includes edge/boundary [10], [11], [3], [1] and contour[4], [15], [19] detection in sensor fields, as well as qualitative/quantitative geosensor field description/monitoring [5], [8]. We classify related work based on whether spatial partition methods are used in their methods.

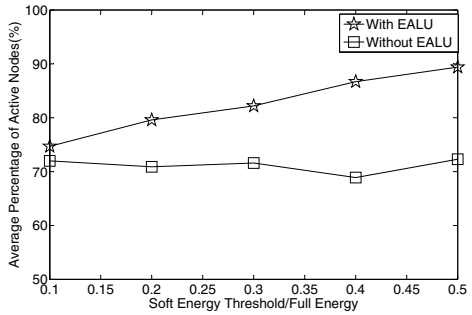


Fig. 9. Effect of Energy Soft Threshold on Percentage of Active Nodes ($r_i = 0.2\,mile, m = 250, n = 25, \theta_s = 0.5\epsilon_f - 0.1\epsilon_f, \theta_h = 0.05\epsilon_f$)

A. Methods Without Spatial Partition

A large class of of methods focus on detecting the boundary sensors without spatial partition. In [3], three alternative schemes are proposed to locally detect edges for the monitored phenomena. A sensor gathers information from its local neighborhood to determine whether it is an edge sensor. The statistical approach uses a boolean function, while the classifier-based approach makes the decision based on classifying incoming data into two subsets. The difference between sensor location and the actual boundary of the phenomena is handled by the "tolerance radius". As such, the continuity of edge sensors are not guaranteed. Therefore a continuous boundary required for region representation may not be directly built upon the edge sensors. By contrast, our work employs spatial interpolation, and use spatial grid partition that can always lead to region representation. Unlike [3], the quality of our detected region is only affected by the resolution of the cell partition, while robust to other parameters. Paper [10] proposes an approach that uses dual space to transform points into line boundaries. The approach only detects convex shapes. Also the transformation is performed on a centralized server. The extension to a localized solution is non-trivial.

The detection and tracking of contours in sensor networks [4], [15], [19] is also related to our work. Contours are continuous curves across the sensor field delineated by the sensors with the same readings. Contours are useful in detecting the formation and dispersion of certain phenomena. However, as [3] pointed out, the notion of interior and exterior is not clear for a contour map. There is no direct match between a contour and a region detected by our approaches.

B. Methods Utilizing Spatial Partition

Another category of methods involves various spatial partition and interpolation methods. A hierarchical boundary estimation algorithm is proposed in [11]. The algorithm starts with

a uniform rectangular partition. The it uses quadtree data structure, and through tree node pruning to obtain a non-uniform partition that approximate the boundary. The algorithm works quite well when sensors are densely deployed. In contrast, our approaches only merge adjacent cells with same properties (e.g. above threshold) and do not approximate. Worboys' work such as [5], [8] employ combinatorial maps and triangulations as the basic elements in describing the qualitative/quantitative change of spatio-temporal fields monitored by geosensors. The paper [13] uses Voronoi cell (VC) to partition the sensors for simple aggregation queries. This automatically implies proximal (nearest neighbor) interpolation. Kriging is used in[6] for sensor field interpolation. The paper [14] applied distributed Kriging algorithm based on its experimental variogram (EV). In our approach, we use spatial grid cells that allows arbitrary linear interpolation functions.

C. Other Related Approaches

The process of selecting cell leaders using our GLS approach resembles the cluster-based routing methods such as [7], where the cluster heads divide the network into several sub-regions, and the nearest head is chosen for data transmission. However, in our approach, the initial set of cell leaders are not randomly chosen as in [7]. Our experiments also show that our approaches outperform such methods of choosing nearest sensors.

Please also note that our work is distinct from the research on optimal sensor placement to achieve maximum coverage and/or minimum communication cost, such as [17], [9]. In practice, the deployment of sensors are often restricted by the natural environment, e.g., the obstacles such as trees or buildings. Therefore an optimal sensor placement cannot always be achieved. Our work is focused on minimizing communication cost based on given sensor placement with known locations.

VI. Summary and Future Directions

In this paper, we have proposed two heuristic approaches to form regions from distributed sensor observations, with the aim of minimizing communication cost. We also proposed an Energy-Aware Local Update model to allow the dynamic reassignment of cell leaders to prevent exhausting the energy of statically selected cell leaders. The experimental results demonstrate that these approaches significantly outperform baseline methods based on random selection and choosing nearest sensors, and are more scalable to the increase of cell resolution and number of sensors.

In our future work, we plan to address more complicated scenarios where multiple sink nodes exist. We will also consider the effect of using more generic interpolation methods. Based on our current work, We also plan to build a prototype system to support queries over the generated regions.

Acknowledgment

This paper is based upon work supported by the National Science Foundation under Grant No. 0844342 and CNS-0709285. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. Banerjee, B. Xie, and D. P. Agrawal. Fault tolerant multiple event detection in a wireless sensor network. *J. Parallel Distrib. Comput.*, 68(9):1222–1234, 2008.

[2] C. Chen, J. Ma, and K. Yu. Designing energy efficient wireless sensor networks with mobile sinks. In *Sensys'06*, 2006.

[3] K. Chintalapudi and R. Govindan. Localized edge detection in sensor fields. In *SNPA '03: the 1st IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[4] K. Dantu and G. S. Sukhatme. Poster abstract: contour detection using actuated sensor networks. In *SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 310–311. ACM, 2003.

[5] M. Duckham, S. Nittel, and M. Worboys. Monitoring dynamic spatial fields using responsive geosensor networks. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 51–60. ACM, 2005.

[6] B. Harrington, Y. Huang, J. Yang, and X. Li. Energy-efficient map interpolation for sensor fields using kriging. *IEEE Transactions on Mobile Computing*, 8(5):622–635, 2009.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020, 2000.

[8] J. Jiang and M. Worboys. Detecting basic topological changes in sensor networks by local aggregation. In *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–10. ACM, 2008.

[9] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *IPSN '06: Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[10] J. Liu, P. Cheung, F. Zhao, and L. Guibas. A dual-space approach to tracking and sensor management in wireless sensor networks. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 131–139. ACM, 2002.

[11] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *IPSN '03: the 2nd ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 80–95, 2003.

[12] N. W. Services. Alert system. http://www.alertsystems.org/.

[13] M. Sharifzadeh and C. Shahabi. Supporting spatial aggregation in sensor network databases. In *GIS '04: Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 166–175. ACM, 2004.

[14] M. Umer, L. Kulik, and E. Tanin. Kriging for localized spatial interpolation in sensor networks. In *Proceedings of the 20th international conference on Scientific and Statistical Database Management*, pages 525–532, 2008.

[15] W. Xue, Q. Luo, L. Chen, and Y. Liu. Contour map matching for event detection in sensor networks. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 145–156. ACM, 2006.

[16] J. Yang, C. Zhang, X. Li, Y. Huang, S. Fu, and M. Acevedo. An environmental monitoring system with integrated wired and wireless sensors. In *WASA '08: Proceedings of the Third International Conference on Wireless Algorithms, Systems, and Applications*, pages 224–236. Springer-Verlag, 2008.

[17] C. Zhang, Y. Zhang, and Y. Fang. Localized algorithms for coverage boundary detection in wireless sensor networks. *Wireless Networks*, 15(1):3–20, 2009.

[18] F. Zhao and L. Guibas. *Wireless Sensor Networks : An Information Processing Approach*. Morgan Kaufmann, ISBN: 1558609148, 2004.

[19] X. Zhu, R. Sarkar, J. Gao, and J. Mitchell. Light-weight contour tracking in wireless sensor networks. In *Infocom '08: The 27th IEEE Conference on Computer Communications*, 2008.