# Distributed Data Aggregation in Sensor Networks by Regression Based Compression

Torsha Banerjee, *Student Member, IEEE* Kaushik Chowdhury, *Student Member, IEEE*
Dharma P. Agrawal, *Fellow, IEEE*
*University of Cincinnati, Cincinnati, OH 45221-0030*
*OBR Center for Distributed and Mobile Computing*
*{banerjta, kaushir, dpa}@ececs.uc.edu*

***Abstract-****In this paper we propose a method for data compression and its subsequent regeneration using a polynomial regression technique. We approximate data received over the considered area by fitting it to a function and communicate this by passing only the coefficients that describe the function. In this paper, we extend our previous algorithm TREG to consider non-complete aggregation trees. The proposed algorithm DUMMYREG is run at each parent node and uses information present in the existing child to construct a complete binary tree. In addition to obtaining values in regions devoid of sensor nodes and reducing communication overhead, this new approach further reduces the error when the readings are regenerated at the sink. Results reveal that for a network density of 0.0025 and a complete binary tree of depth 4, the absolute error is 6%. For a non-complete binary tree, TREG returns an error of 18% while this is reduced to 12% when DUMMYREG is used.*

**Keywords:** Aggregation, Attribute-based Trees, Dummying, Function-approximation, Polynomial Regression

## I. Introduction

Wireless sensor network applications require sending of huge amounts of relevant data from one point of the network to another. This necessitates a fast and robust data aggregation protocol which performs data compression without substantial loss in accuracy, addresses considerations of storage and facilitates quick retrieval of attributes. Like most physical attributes, sensed parameters exhibit a gradual and continuous variation over 2D Euclidean space. The basic idea of our scheme is as follows: Attribute values show a smooth spatial gradation, *i.e.* there is a correlation between attribute values and location as argued in [2], and between multiple sensors in close proximity [3]. *DUMMYREG* leverages this phenomenon by aggregating the correlated attribute values from sensors and eliminating redundancy in the process. In case the tree is not complete, the parent approximates the information for the missing child based on the data available from the other child. In cases where neither child is present, the parent uses its own data for the approximation process. Our scheme first creates attribute-based non-complete binary tree (or query tree) called NQT and applies the probabilistic bound with which a node joins the tree based on network density derived in [19]. Our approximation algorithm performs better with increasing children at each level and hence we consider binary trees as the worst case scenario to show performance improvement. After the tree construction phase, sensing nodes report the sensed values to the tree nodes closest to them. Each tree node then calls the regression function and obtains the coefficients, $(\beta_0,.....,\beta_8)$ which is then passed to the higher level instead of raw data. Thus nodes at each level use the coefficients of their children to improve the approximation function and this procedure stops at the root. The sink now has access to an approximation, $f(x, y)$, of the sensed attribute at any point in the region spanned by the tree. These values can be obtained by choosing suitable $x$ and $y$ co-ordinates. The rest of this paper is organized as follows: Sec. II lists the preliminaries and discusses the most prevalent existing work in this area. In Sec. III, we summarize the root selection algorithm proposed in [19] for better understanding of our work. This section also proposes the modified *TREG* [19] algorithm called *DUMMYREG* for non-complete binary trees. Simulation results are presented in Sec. IV. Finally Sec. V concludes the paper.

## II. Related work

The existing work in data aggregation in wireless sensor networks is studied in this section. In LEACH [5], a set of nodes are selected randomly as clusterheads (*CH*s) and each node joins a cluster depending on the communication energy between the node and the *CH*. The role of *CH* also keeps changing to preserve energy. However, the limitation of this scheme is that the *CH*s themselves may run out of energy to transfer data to the base station as there are only few nodes which act as *CH*s and the sink is assumed to be located far away from them.

In [6], a node is elected as the representative node for sending the snapshot of a sensed region to the base station. Though this scheme reduces the overall number of nodes required in a query request made by the sink, it nevertheless involves the election algorithm to be run at definite intervals of time (making it strictly proactive) keeping in mind the error threshold to be satisfied. Our proposed scheme avoids this extra work by selecting a random node location (the node might not be actually present) as the representative of the sensed region in a reactive manner unlike [6].

In Greedy aggregation-tree approach [7], a shortest path is built between the first source to the sink and subsequent sources connect to the closest nodes of the existing tree by creating incremental least energy paths. This scheme returns high savings for proactive systems. However, for attribute based queries [8], in which the sink has to query for attribute values in a certain range, a network-wide flooding is used which is costly in terms of communication energy. Also, sink mobility is not supported as gradients, once set up, are unchanged during the operation of the scheme.

In the compression technology proposed in [9], a base signal needs to be updated. The collected data is partitioned into intervals for approximation. Our proposed regression algorithm is simpler in this regard as no pre partitioning of data is necessary thus having substantially less overhead. Since, data is not filtered before the actual compression process, bandwidth required may be more compared to [9], however a trade-off is achieved between consumed bandwidth and latency of the compression process. Unlike [9], our scheme does not involve comparing the error incurred after every update process to a predefined error threshold. Nevertheless, error incurred after running our algorithm is less than 6% for complete binary aggregation tree.

Distributed kernel regression [12] has similarities with our scheme with the following important differences. In the former, each node sends a message containing a square matrix and a vector (to summarize the measurements over its local region) the size of which increases with the number of neighbors with which it shares kernel variables, thereby increasing the energy consumption in the communication. In our scheme, the data packet sent by each node is constant, comprising a set of coefficients and boundary of the area from which values have to be re-generated to continue the aggregation process.

## III. Our proposed scheme

Our scheme considers a distributed scenario where each subregion corresponds to an aggregation tree. Thus each subregion approximates attribute values through distinct set of coefficients. We form disjoint attribute-based trees i.e. NQTs, the root of each being decided by our DECIDE_ROOT algorithm [19] in a distributed manner. The nodes of NQT do not sense any data and participate only in the compression process. We now summarize the DECIDE_ROOT algorithm in the next sub-section.

### A. Decision of a node to become the root

The root selection method is discussed in details in [19] and the main points are mentioned in this paper. The root in each subregion is selected in such a manner so that minimum routing is required among them when the sink needs attribute information involving more than one subregion. For a node at $(X_a, Y_a)$, DECIDE_ROOT and the function ROOT called by it, first enables it to identify if it lies within the permissible distance from the y axis of the optimal root location. This is repeated for the x axis, and any node which satisfies both these conditions is eligible to be the root and all such nodes broadcast their eligibility.

**PSEUDOCODE** ROOT(Max, value)

```
Input: the maximum value of the coordinate in the network, corresponding
coordinate of the node
Output: Boolean value true or false depending on whether the node will be the root
or not respectively..
begin
ans=false
n=1
find minimum n such that
value>(Max-(n-1)×l)     // (where l is the length of a side of the subregion)

if n is even then
   if |Max − n × l − value| < δ then
        ans=true
end
```

**PSEUDOCODE** DECIDE_*ROOT* ($X_{max}$, $Y_{max}$, $X_a$, $Y_a$)

```
Input:  the maximum x-y coordinates of the network,
        coordinates of the node
begin
   if d/l is even   //(where d is the length of the smallest square defining a sensing
region)
       root_y=call ROOT(Ymax, Ya)
       root_x=call ROOT(Xmax, Xa)
   else
       if Ya < l then
           if |l − Ya| < δ then root_y=true
           else root_y=call ROOT(Ymax, Ya)
       if (Xa > Xmax − l) then
           if |Xmax − l − Xa| < δ then root_x=true
           else root_x= call ROOT(Xmax, Xa)
end
```

### B. Discussion on Query tree

Unlike [19], in this paper we consider a more general case where query trees (binary) can be formed consisting of nodes with less than two children. This is possible when nodes can be out of range of each other after random deployment. The trees are of a pre-assigned depth, $p$ and hence involve a maximum of $p$-hops for complete traversal. All the nodes of a tree store the same attribute type. We now derive an upper bound on the depth $p$ of an NQT given the area of the network $A$ and the total number of nodes in the network, $D$. Density ($\rho$) of the network is $D/A$. $A_s$ is the area of a sub region, which contains a single compression, tree $T_c$. Therefore, average number of nodes, $S$, in the subregion is given by $\rho \times A_s$. For a non-complete binary tree for our case, let the total number of nodes be $t$ where

$$t = ceil((2^{(p+1)} - 1)/2 + 1) \qquad (1)$$

Again, assuming that $n_s$ is the average number of sensing nodes reporting to each tree node,

Lower bound on $S = n_s \times t + t$ *or* $t = S/(n_s + 1)$

Substituting the value of $t$ in Eq. (1); we get an optimal value for the depth of query tree (NQT), $p = \ln(\frac{2 \times S - n_s - 1}{n_s + 1}) - 1$

As an example, $D = 1630$, $A = 800 \times 800$.

$\therefore \rho = 0.0025$, $A_s = 400 \times 400$.

Upper bound on the number of nodes in the region $S = 0.002469 \times 400 \times 400 = 408$.

Assuming depth $p=4$ and number of nodes in $T_{c,t=17}$, where $n_s=12$. Thus, the total number of sensing nodes= $17 \times 12 = 204$ and hence the number of nodes actually in the region $S = 17 + 204, = 221 < 408$.

Thus, the parameters defined above are valid.

By ensuring that NQTs are spread along the entire network, attribute readings sent by the sensing (NT) nodes to the corresponding NQT incur a smaller hop count. Every time a node has to select two of its children, it selects the two nodes farthest apart. This ensures that the tree is widely spread, covering as much sensing region as possible so that the maximum accuracy of the approximation process achieved. This is asserted by running the algorithm, *FORM_QT* [19]. It also ensures that redundancy in reported attribute values is reduced.

## C. Function approximation

Each node of the query tree stores the attribute value sent by each of the nearest non-tree (NT) sensor nodes. These NT nodes report their data to the tree node closest to them, for storage of the current attribute reading. Recall that NT nodes only sense attributes while the NQT is for storage only. Any arbitrary tree node $i$ of a NQT creates a function approximation $f_i(x, y)$ from all the data tuples of the form $(z_i, x_i, y_i)$ stored in $i$. Using multivariate polynomial regression, a polynomial equation is generated with three input variables (z, x, y) for all the data points in one particular node of NQT. A general multilinear regression model is as follows [11]:

$$y = f(x_1, x_2, ...., x_m) = a_0 + \sum_{k=1}^{m} a_k x_k$$

where $x_1, x_2, ..., x_m$ are the independent variables called predictors of the model and y is the dependent variable. The observations are sampled and the observed values of the vector variable y are used at the particular levels of $x_k$ to estimate $a$. $y$ is the n-element vector of sample values and $\vec{a}$ is the (m+1)×1 vector estimate of $a$.

Applying least-square criterion, the squared error needs to be minimized i.e.

$$F(\vec{a}) = (X\vec{a} - \vec{y})^T (X\vec{a} - \vec{y})$$

Where

$$X = \begin{pmatrix} 1 & x_{11} & x_{21} & .. & x_{m1} \\ 1 & x_{12} & x_{22} & .. & x_{m2} \\ : & : & : & : & : \\ 1 & x_{1n} & x_{2n} & .. & x_{mn} \end{pmatrix} ...(2) \quad \vec{y} = \begin{pmatrix} y_1 \\ : \\ : \\ y_n \end{pmatrix} ...(3) \quad a = \begin{pmatrix} a_0 \\ a_1 \\ : \\ a_m \end{pmatrix} ...(4)$$

Necessary condition for a minimum is that the partial differentiation of $F(\vec{a})$ w.r.t $\vec{a}$ is zero.

i.e., the normal equation obtained is $X^T X\vec{a} = X^T \vec{y}$ (5)

The system has a solution if $X^T X$ is not singular i.e. it has an inverse. Therefore multiplying both sides of (5) by $(X^T X)^{-1}$ we get $\vec{a} = (X^T X)^{-1} X^T \vec{y}$, where $(X^T X)^{-1} X^T$, called the Pseudoinverse of the matrix X is a generalization of the inverse $X^{-1}$. Using polynomial regression for our model, we get the following equations analogous to relations 2–4.

$$X = \begin{pmatrix} 1 & y_1 & y_1^2 & x_1 & x_1y_1 & x_1y_1^2 & x_1^2 & x_1^2y_1 & x_1^2y_1^2 \\ 1 & y_2 & y_2^2 & x_2 & x_2y_2 & x_2y_2^2 & x_2^2 & x_2^2y_2 & x_2^2y_2^2 \\ : & : & : & : & : & : & : & : & : \\ 1 & y_n & y_n^2 & x_n & x_ny_n & x_ny_n^2 & x_n^2 & x_n^2y_n & x_n^2y_n^2 \end{pmatrix} z = \begin{pmatrix} z_1 \\ z_2 \\ : \\ z_n \end{pmatrix} \text{ and}$$

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ : \\ \beta_8 \end{pmatrix} \text{ where } \vec{\beta} = (X^T X)^{-1} X^T \vec{y} \quad (6)$$

$$p(x,y) = \beta_0 + \beta_1 y + \beta_2 y^2 + \beta_3 x + \beta_4 xy + \beta_5 xy^2 + \beta_6 x^2 + \beta_7 x^2 y + \beta_8 x^2 y^2 \quad (7)$$

When $\vec{\beta}$, obtained from relation 6, is used with a given location (x,y), we solve z = p(x,y) to retrieve the attribute value at a node location (x,y). Note that p(x,y) is the final function available at the root. For estimating β, a unique inverse of X should exist i.e. $X^T X$ must be of full rank $m+1$ [10], given that $\vec{\beta}$ is a $(m+1)\times 1$ vector. In other words, $n>>m+1$ and no column of X can be expressed as weighted linear combination of any set of other columns. Based on the function approximation process, we have proposed the algorithm, *DUMMYREG* stated below. Inputs to the algorithm are the depth and the number of sensing nodes reporting to each tree node. Algorithm, *DUMMYREG* is a modified version of *TREG* where spatial correlation of attribute values is taken into account to create readings at locations devoid of actual aggregating nodes. A parent node $k$ regenerates readings by regenerating random node locations in the virtual area spanned by each of its virtual children, $i$ or/and $i+1$. In cases II and III, coefficients of the real child (the child node present) are used to regenerate attribute readings for the virtual child (the child node absent). In case IV, since the parent node has no children, therefore, it uses its own coefficients to regenerate readings for its two virtual children. This method of dummying attribute readings increases the accuracy of the overall

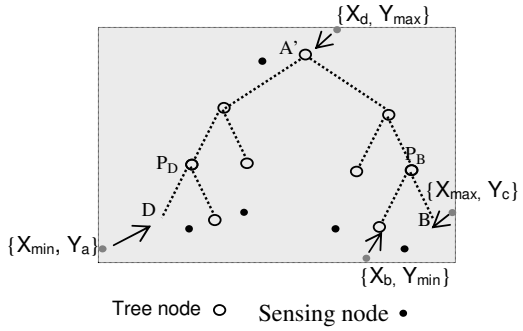


Fig. 1. Illustration of how a node calculates the boundary of the region for data regeneration

compression process by including readings from regions devoid of actual sensor nodes. Without any dummying i.e. without the inclusion of attribute values from non-existent nodes in the overall approximation process, error incurred is the highest. This is evident from the fact that readings that would have otherwise come from the region spanned by the non-existent nodes are not considered in the compression process. With partial dummying, a parent node follows case IV of *DUMMYREG* to regenerate attribute values of both of its non-existent children. But, regeneration is not done for the case when one child is present. For the full dummying case, the error incurred is minimum as attribute values are included from the entire region irrespective of whether a node is actually present or not. The lower limit of the x-coordinate (xlow) of the left child of a parent node *k* and the upper limit of x-coordinate (xhigh) of its right child is assumed to be *u* (a pre specified system parameter which depends on the size of the network) units each. When node *k* does not have any children, the x-coordinates of its children are approximated to lie within *u* units of the midpoint of the area spanned by its sensing region. Their y coordinates are assumed to be *u* units below that of the parent. Assuming node *k* to be the parent of nodes *i* and *j*, each of nodes *i* and *j* uses relation 6 to generate the coefficient tuple $(\beta_{i0},.....,\beta_{i8})$ and $(\beta_{j0},.....,\beta_{j8})$ respectively and sends this set to node *k*. Node *k* now generates two sets of random (x, y) locations and calculates the corresponding values of the sensed attribute at each such location by using the coefficients sent by their children. These two data sets are then appended with *k*'s own reported readings to calculate the new set of coefficients that will be passed to *k*'s parent at the next higher level. This process is continued until the root node is reached which will have the final set of coefficients to be used by the sink. In this process, it is important to identify the region over which the data values are generated as they directly affect the accuracy of the approximation. We identify this region as the area bounded by the coordinates $\{x_{\min,}\, y_{\min}, x_{\max}, y_{\max}\}$, where the minimum and maximum are taken over all the sensing nodes in the subtree under the current parent that report to tree nodes.

**PSEUDOCODE** DUMMYR*EG* (***p***, $n_s$)

**begin**

1. **for** each of the leaf nodes i of the tree
   a. File node"i".dat is read
   b. Multivariate polynomial regression is performed on each data file and the coefficients are stored in the each of the arrays $\beta_0, \beta_1, .....\beta_8$ each of size N.
   **endfor**
2. Initialize *level* to $2^p$
   **while** *p* is greater than *0*
   a. *sum*= $level + 2^{p-1}$
   *b.* *fk=level*
   *c.* **while** *k<sum*
   **for** each of the non leaf nodes k of the tree,

`

**Case I:** *k* has 2 children
k computes random x-y points for each of its 2 children *i* and (*i+1*) where (*xmin, ymin*) and (*xmax, ymax*) are the coordinates of the leftmost and down most node and rightmost and top most node respectively reporting to the nodes *i* and *i+1*.

**Case II**: *k* has only node i as its child and *i* even (*i* is left child), it computes random x-y points for *i+1* where (*xmin, ymin*) and (*xmax, ymax*) are the coordinates of the leftmost and down most node and rightmost and top most node respectively reporting to node *i+1* It sets $(\beta_{(i+1)0},.....,\beta_{(i+1)8})$ to $(\beta_{i0},.....,\beta_{i8})$ to regenerate readings for node *i+1*.

```
xlow[i+1]=xhigh[i];
xhigh[i+1]=xhigh[k]+u;
ylow[i+1]=ylow[i];
yhigh[i+1]=yhigh[i]
```

**Case III:** node only i+1 is present and *i+1* is odd (*i+1* is the right child), it computes random x-y points for i where (*xmin, ymin*) and (*xmax, ymax*) are the coordinates of the leftmost and down most node and rightmost and top most node respectively reporting to the node *i*. It sets $(\beta_{i0},.....,\beta_{i8})$ to $(\beta_{(i+1)0},.....,\beta_{(i+1)8})$ to regenerate readings for *i*.

```
xlow[i]=xlow[k]-u;
xhigh[i]=xlow[i+1];
ylow[i]=ylow[i+1];
yhigh[i]=yhigh[i+1],
```

**Case IV**: k has no children; k performs regression on its reported values and generates coefficients, $(\beta_{k0},.....,\beta_{k8})$. It sets $(\beta_{i0},.....,\beta_{i8})$ and $(\beta_{(i+1)0},.....,\beta_{(i+1)8})$ to $(\beta_{k0},.....,\beta_{k8})$ to regenerate readings for each of its non-existent children.

```
cc=yhigh[k]-ylow[k];
xlow[i]=xlow[k];
xhigh[i]=Ceiling[(xlow[k]+xhigh[k])/2];
ylow[i]=ylow[k]-cc;
yhigh[i]=ylow[k];
xlow[i+1]=Ceiling[(xlow[k]+xhigh[k])/2];
xhigh[i+1]=xhigh[k]+u;
ylow[i+1]=ylow[k]-cc;
yhigh[i+1]=ylow[k];
```

**endfor**

Using $(\beta_{i0},.....,\beta_{i8})$ and $(\beta_{(i+1)0},.....,\beta_{(i+1)8})$, new attribute values are calculated and appended to node"k".dat. Node k then calls the regression function to calculate $(\beta_{k0},.....,\beta_{k8})$ and passes it to its parent.

**endwhile**

*level=sum*

**endwhile**

**end**

As an e.g., in Fig. 1, consider *A'* as the current aggregating node. The shaded area represents the region in which data values will be re-generated by *A'*. This region is bounded by the minimum and maximum co-ordinates of the sensing nodes (colored gray) reporting to the subtree under *A'*. *A'* gets the boundary coordinates of this region from its children. Here, nodes *B* and *D* are not present and therefore they are the dummy nodes. However, the area bounded by them is computed from their neighbors' $\{x_{\min,}\, y_{\min}, x_{\max}, y_{\max}\}$ ranges using *DUMMYREG* algorithm. Thus *A'* still covers the entire area of the subregion under it thus providing accurate approximation process though some nodes are not actually present in the subregion. $P_D$, parent of node *D* follows case II of *DUMMYREG* to regenerate virtual attribute readings at *D*. Similarly, $P_B$, parent of node *B* follows case III of *DUMMYREG* to regenerate virtual attribute readings at *B*.

When a sink needs to know the attribute value at a particular location (*x, y*), it sends the query of the form "SELECT *attribute* FROM *sensors* WHERE *location*= (*x, y*)" to the root. The query is first propagated down the NQT to reach the leaf nodes at the last level. The aggregation model follows a bottom-up approach. Reporting data through the aggregation process should be much more efficient in terms of energy and latency, than sending individual data bits corresponding to a specific geographical co-ordinate. To prove our claim, a parameter called compression ratio is defined as the number of bytes transmitted in NQT after compression to the original number of bytes transmitted in NQT. Compression ratio (based on a round of data aggregation) is calculated as follows: Assume that *NQT* is a non-complete binary tree of depth *p* of atmost $2^p$ leaf nodes. Each attribute packet of size $s_i$ bytes contains the attribute reading and the coordinates of the location where the reading is taken. Therefore, the number of bytes input to each leaf node is only this data of size $n_s \times s_i$, since $n_s$ is the total number of sensing nodes reporting to each tree node.

For simplicity of understanding, we have considered a specific case of non-complete binary tree which has $\left((2^{p+1}-1)/2+1\right)$ nodes.

Apart from the attribute readings from the $n_s$ sensing nodes, each non leaf node gets as input from its children, the coefficients and the x-y boundaries of the area to be regenerated.

Thus, $T_{nl}$ (the total number of bytes input to the non-leaf nodes)= $(n_s \times s_i + 2 \times (s_x + s_y + s_c)) \times \left((2^{p+1}-1)/2+1-2^p\right)$

The output packet of size $(s_x + s_y + s_c)$ bytes from a tree node contains the coefficients and the x-y range.

The total number of bytes output from all the nodes can be given as $T_o = (s_x + s_y + s_c) \times \left((2^{p+1}-1)/2+1\right)$

Upper bound on the compression ratio, *CR* (output : input size) can be obtained as

$$= \sum_{alledges} \frac{no.of\ bytes\ transmitted\ on\ an\ edge\ after\ compression}{no.of\ input\ bytes\ on\ an\ edge}$$

$$= \frac{T_o}{T_l + T_{nl}} = \frac{(s_x + s_y + s_c) \times t}{(0.5 * n_s \times s_i + (s_x + s_y + s_c))}$$

where *t* is the total number of tree nodes given in Eq. (1).

The success of the function approximation lies in sending only 8 bits as coefficients instead of actual data that is of much higher size. The 8 bits correspond to a polynomial in x and y of degree 2 that is sufficient to approximate the smooth contour of spatially correlated sensor readings and a polynomial of higher degree is not required.

## IV. Simulation Results

We first create the aggregation trees applying our proposed TREECAST algorithm [19] and using Simjava as the simulator [16]. We assume a collision free MAC protocol and list the simulation parameters and their definitions in Table 1. The main focus of our paper is on the performance evaluation of our proposed *DUMMYREG* algorithm. *Mathematica* [18] tool is used for calculating the coefficients at each node during the execution of this algorithm. While evaluating the results we consider the following metrics: (1) compression ratio (2) contour matching between real and approximated data for a snapshot of the region approximated (3) the accuracy of the approximation of the sensed parameter over the entire region both in absolute value and percentage error. Our simulation study consists of two different data sets: a synthetic model generated in the laboratory based on spatial correlation of attributes and real world data model taken from the data set from rooftop of ATG, University of Washington [17] (relative humidity, temperature). We first generate a temperature gradation contour over a 2-D region for testing the accuracy of our algorithm as shown in Fig. 3(i). The temperature attribute shows a change of 1 unit for a traversal of every 45 units. We then place about 400 sensor nodes in an area of $400 \times 400$ square units. Over this area, a temperature range of $30^\circ - 34^\circ F$ can be realistically assumed and validated by a smaller temperature variation shown in National Observational Data [15] for an undisturbed region.

Table I
VALUES OF SIMULATION PARAMETERS USED

| SYMBOL | DEFINITION | VALUE |
|---|---|---|
| *A* | Area of the network | 800*800 |
| *R* | Radio Range | 40m |
| *D* | Total number of nodes in the network | 1630 |
| ρ=*A/D* | Density of the network | 0.0025 |
| *As* | Area of a sub region having a tree | 400*400 |
| *p* | Depth of the query tree | 4 |
| $n_s$ | Avg. no. of nodes reporting to each tree node | 12 |

1. Compression ratio: Fig. 2 shows the variation of compression ratio with depth of the tree and as expected it is found to be almost constant giving a value of 0.0005. The descending nature of the curve in Fig. 4 suggests that with increase in depth, compression ratio expressed as the output data size as a fraction of the input data size, decreases. This is expected as greater is the depth, better is the degree of compression as further reduction in output data occurs. A high rate of compression reduces the overall message size thus saving the overall communication bandwidth and energy. Also, a constant compression ratio is preferred as it makes it easier to model the hierarchical structure of the query tree.

2. Contour Plot: In Fig. 3(i) our synthetic model depicts the correlation between sensed attribute values and coordinates of NT nodes. Fig. 3(ii) depicts the same but with approximated values obtained after running *DUMMYREG* on each NQT. In this plot we measure temperature in Fahrenheit scale.

A comparison of the plots shows that our scheme does not limit the accuracy to certain regions of the contour. The gradation and scales of approximated temperature contour matches the actual temperature distribution in the same region almost exactly.

3. Percentage Error: Fig. 4 shows the variation in percentage error with depth of both complete and non-complete binary trees. The white bars show the error incurred when TREG is applied on complete binary tree of varying depths. The best case error is 5.62%. Black bars indicate the errors incurred when *DUMMYREG* is applied to non-complete binary trees of varying depth. In this case the best case error is 12.1%. The gray bars show the corresponding errors (best case error is 18%) when TREG is applied to non-complete binary trees. Thus we see that without the dummying of nodes, the entire region is not approximated properly giving greater error that the case of full dummying of the sensed region. For each of the cases considered, we observe a steady fall in mean error and percentage error for synthetic data as the depth of the query tree increases. This is expected, as with increase in depth, the tree nodes are able to cover larger area thus being able to approximate the sensed parameter over the region better. A tree depth of 4 is considered to be optimal as all best case errors occur at this level. A NQT of depth 4 gives error which is reasonably below error threshold (6% for a complete binary case) and at the same time produces less latency in the overall aggregation as number of approximate steps are less (it increase proportionately with increase in tree depth). A tree with depth 1 gives maximum error, as a small tree of only three nodes (most sensing nodes are dispersed in the region, not within range of tree nodes and are hence unable to report to the NQT) is made to approximate the entire region of area *As*.

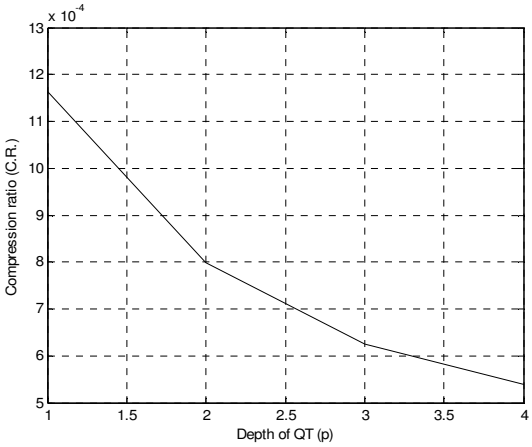


Fig. 2. Dependence of compression ratio on depth of NQT

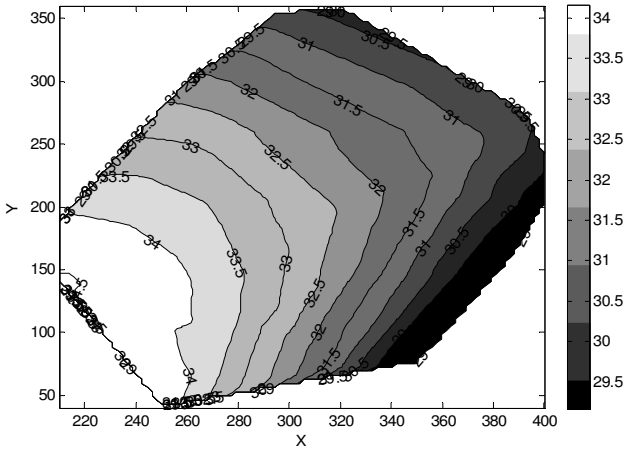


Fig.3(i). Contour plot of our synthetic model in a 400×400 subregion

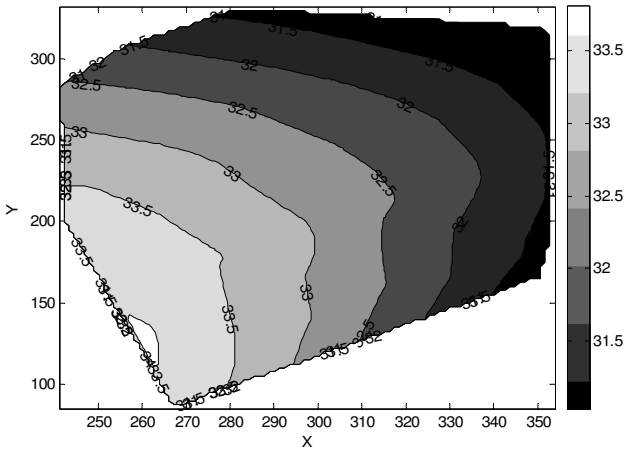


Fig. 3(ii). Contour plot of the attribute in the same region obtained after simulation

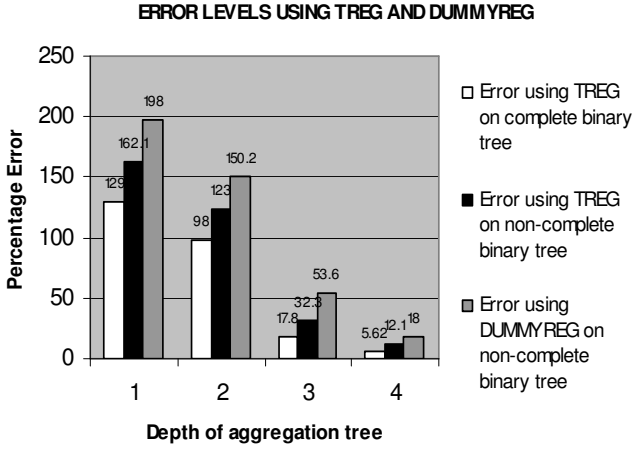


Fig. 4. Variation of percentage error with depth and tree type

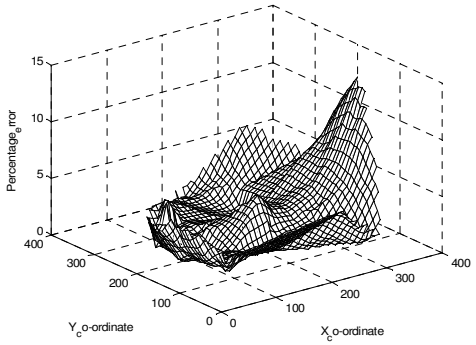


Fig.5. 3-D error plot for depth 4

In this paper, we have calculated the error incurred in the approximation process for a realistic case i.e. a non-complete binary tree. Error is calculated as the absolute deviation from the true value and percentage error

$$E = \left( \frac{| z - \bar{z} |}{z} \times 100 \right)_{Th}$$

For a depth of 4 of the non-complete query tree, we obtain different error levels respectively for three cases (i) no dummying, (ii) partial dummying and (iii) full dummying. From Table II, we observe a steady fall in percentage error as different levels of dummying are applied. Without dummying, readings from locations without tree nodes do not participate in the *DUMMYREG* algorithm. By dummying the otherwise absent tree nodes, imaginary attribute values are generated and made to participate in the compression method thus approximating readings from the entire sensed region instead of only the area where sensors are actually present. Thus, when attribute values are regenerated at the root spanning the entire region, the non-dummying case produces attribute values at some locations highly uncorrelated form the true values since readings from these locations were not considered by the parent nodes due to absence of sensors (children nodes) there. But, the dummying case gives better results when values are regenerated back at the root as readings from the entire spanned region were incorporated in the approximation process. This reduces the error level considerably, improving our approximation scheme in terms of accuracy.

4. 3-D error plot: Fig. 5 shows 3-D percentage error plot for NQT of depth of 4. The error plot has a concave shape with peaks at the border. This is because the tree structure is concentrated at around the central areas [19] of the sub-region and nodes at the extremities do not report to the query tree. The approximation at the edge of the region is greater and this results in higher errors as is shown in the plot.

Table II
ERROR LEVELS INCURRED FOR DIFFERENT CASES OF ATTRIBUTE REGENERATION

| CASES | ERROR LEVEL |
|---|---|
| No dummying | 18% |
| Partial dummying | 15% |
| Full dummying | 12% |

## V. Conclusions and future work

In this paper, we propose *DUMMYREG*, an extension to TREG [19] algorithm, a novel data aggregation strategy by building distributed multiple attribute-based trees based on function approximation. Similar to the idea put forward in [6], the key assumption of our paper is that a node can replace another node in a query when their readings collected are similar in the quantitative sense. This scheme allows inclusion of attribute readings at locations devoid of sensor nodes thus increasing the overall accuracy of the approximation process. Like [19], this scheme also provides constant message size irrespective of the depth of the aggregation tree. Our scheme is scalable as the data compression ratio is nearly constant and accuracy increases with increasing density. Future work includes computing the energy required for communication and latency involved per round of data aggregation. Analysis of the approximation polynomial needs to be done to compute local maxima and minima of error in attribute values.

## REFERENCES


[1] D. Agrawal and Q-A. Zeng, *Introduction to Wireless and Mobile Systems,* Brooks/Cole Publishing, 436 pages, ISBN 0534-40851-6, 2003.

[2] E. Kuh, An Essay on Aggregation Theory and Practice.

[3] Mehdi Sharifzadeh and Cyrus Shahabi, "Supporting Spatial Aggregation in Sensor Network Databases," *GIS'04*, November 12–13, 2004.

[4]S. PalChauduri, S. Du, A. Saha and D. Johnson, "TreeCast: A Stateless Addressing and Routing Architecture for Sensor Neworks," 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks, *WMAN* 04.

[5]W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks,"' *Proc of the 33rd Hawaii International Conference on System Sciences (HICSS '00)*, January 2000.

[6] Yannis Kotidis, "Snapshot Queries: Towards Data-Centric Sensor Networks*," ICDE*, 2005.

[7] C. Intanagoniwiwat, Estrin and Govindan., "Impact of network density on data aggregation in wireless sensor networks," *Proc. of the 22nd International Conference on Distributed Computing Systems*, July 2002, pp. 575-578.

[8] Bhaskar Krishnamachari, Deborah Estrin and Stephen Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks," *IEEE Infocom*, 2002.

[9] Antonios Deligiannakis, Yannis Kotidis and Nick Roussopoulos, "Compressing Historical Information in Sensor Networks," *Proceedings of ACM SIGMOD* 2004.

[10] Jeremy D. Finn, "A General Model for Multivariate Analysis," ISBN 0-03-083239-X.

[11] Christian Borgelt, "Intelligent Data Analysis," School of Computer Science Otto-von-Guericke-University of Magdeburg,Universit¨atsplatz 2, D-39106 Magdeburg, Germany.

[12] Carlos Guestrin, Peter Bodix, Romain Thibaux, Mark Paskin and Samuel Madden, "Distributed Regression: an Efficient Framework for Modeling Sensor Network Data," *IPSN '04.*

[13] Wei Yuan, Srikanth V. Krishnamurthy, Satish K. Tripathi, "Synchronization of Multiple Levels of Data Fusion in Wireless Sensor Networks," *IEEE GLOBECOM,* 2003.

[14] Ignacio Solis and Katia Obraczka, "In-Network Aggregation Trade-offs for Data Collection in Wireless Sensor Networks," *INRG Technical Report 102*, 2003.

[15] ATG [Online]
www.atmos.washington.edu/data/difax/maxtemp

[16] Simjava [Online]
www.dcs.ed.ac.uk/home/simjava/doc/index-all.html

[17] Tilman Wolf and Sumi Y. Choi, "Aggregated Hierarchical Multicast for Active Networks," *MILCOM* '01.

[18] Mathematica [Online] www.wolfram.com

[19] Torsha Banerjee, Kaushik Chowdhury, D.P. Agrawal, "Tree Based Data Aggregation in Sensor Networks Using Polynomial Regression," *Fusion* 2005.