Tracking Mobile Targets Using Grid Sensor Networks

Ahmed M. Khedr
Mathematical Department, Faculty of Science,
Zagazig University, Egypt
email amkhedr@yahoo.com

Abstract

We have considered a sensor network where a lot of sensor nodes are spread in a grid like manner. These sensor nodes are capable of storing data and thus act as a separate dataset. The entire network of these sensors act as a set of distributed datasets. Each of these datasets has its local temporal dataset along with spatial data and the geographical coordinates of a given object or target. In this paper an algorithm is introduced that mines global temporal patterns from these datasets and results in the discovery of linear or nonlinear trajectories of moving objects under supervision. The main objective here is to perform in-network aggregation between the data contained in the various datasets to discover global spatio-temporal patterns; the main constraint is that there should be minimal communication among the participating nodes. We present the algorithm and analyze it in terms of the communication costs.

Keywords: Wireless sensor networks, in-network aggregation, spatio-temporal patterns, distributed datasets, data mining.

1 Introduction

Sensors are tiny electronic devices equipped with a battery for energy source, sensing module for sensing physical characteristics, an onboard processor for performing computations, a wireless transceiver for two way communications with other sensors and sufficient memory for storing local sensing variables and for local computations. A sensor network consists of a set of such sensors, deployed in hundreds to thousands over a region, that is capable to gather acoustic, magnetic, spatial, or seismic data and performing distributed computations over the gathered data by individual sensors to make meaningful inferences and then send the data to end user or base station.

Wireless sensor networks promise novel applications in several domains. Forest fire detection, battlefield surveillance, or monitoring of human physiological data are only in the vanguard of plenty of improvements encouraged by the deployment of sensor networks. Sensor nodes can be spread out in dangerous or remote environments whereby new application fields can be opened. Wireless sensor networks enable the monitoring of a variety of possibly inhospitable environments that include home security, machine-failure diagnostic, chemical/biological detection, medical and wild habitat monitoring. One of the main challenges raised by sensor networks is the fact that they are usually power constrained, since sensing nodes typically exhibit limited capabilities in terms of processing, communication, and especially, power. Sensor networks'power limitation is buildup by the fact that, often, once deployed; they are left unattended for most of their lifetime. Thus, energy conservation is of prime consideration in sensor network algorithms in order to maximize the network's operational lifetime.

In our paper, we assume that stand alone sensors are spread in a grid like manner in a region to be supervised. Every sensor is capable of collecting information in its vicinity and exchanging it with its neighboring sensors. Both the sensing range and the distance from the neighbors are customizable attributes. The sensors are capable of running self-decomposable algorithms that extract useful information from the data stored in the sensors. Upon collection of data, the main interest to us is to determine global patterns from these sensors. Mobility of the sensors is dependent on the application and its use. We have restricted our paper work to stationary sensors.

Our main aim is to perform data mining using our algorithm on the data stored in these sensor nodes. Data mining in sensor networks has a host of real time applications that can yield very useful and profitable results. For example, the following questions can be answered by implementing Data Mining techniques on the vast data available to us from a set of distributed databases in a state. Is there a pattern of increasing crime rate in the different counties of the state? How is this pattern spreading over the state and in which direction? Which were the most affected areas in the state by hurricane Isabel? Which areas are most likely to be affected by a power outage? Some of the main objectives for designing and developing algorithms for mining of distributed data are: 1) There should be minimum communication between the different data sites; 2) The communication among the data sites should be secure.

Traditional data mining depends on centralized data in that the central site obtains and processes the compressed information from all the sensors. Each sensor could report a time stamp and other measurable values, without the need for detailed measurements. The central site receives such information from all sensors and analyzes it to arrive at useful information like whether there is a pattern in the sensors or whether any sensors are reporting outlier values (which might indicate defective sensor nodes). But in such a case, bandwidth limitations make it virtually impossible to accumulate all sensor data at a central location for processing. Exchange of information between nodes and the central processing node would result in a very high communication cost of the network. Since communication cost is a major constraint, it is not practical to transfer and integrate large amounts of data to a single site before carrying out an analysis. This only results in very high overall complexity of the system. In most of the sensors, the local computations cost is very low when compared to the communication cost between sensors. A lot of local processing capabilities is present in each of the sensor nodes. Hence we need to develop algorithms which would minimize the exchange of messages between sensors and maximize the computations at the local site. The main aim of the algorithm would be to minimize the use of the available bandwidth and to maximize the use of the local processing sensor node site. Thus, instead of the centrally processing all data, algorithms need to be designed to summarize and aggregate data while they are in the network.

Our algorithm development based on the analysis of location-time points for patterns in sensor network, one of the interesting applications of our algorithm is mining the trajectories of animals in a farming area, to determine migration patterns of certain groups of animals. To the best of our knowledge the use of location-time point's analysis for patterns discovery in sensor network has not been explored before.

2 Related Research

There are many technical challenges associated with sensor networks, such as self-organizing algorithms, energy-efficient routing protocols, data analysis/mining technology and network lifetime improvements [2, 11, 16, 24]. The source of energy for a wireless sensor is most often an attached battery. The power in sensor nodes can be used up simply by computations and transmissions. Furthermore it is infeasible to replace thousands of nodes in hostile or remote regions. Therefore, conserving energy so as to prolong the network lifetime is becoming one of the key challenges for such power constrained networks. Recent research has addressed this topic for example [6, 14].

The term in-network processing denotes the data processing that happens inside a network. It is usually used for data aggregation, collaborative signal processing and other similar problems in which data is required from a majority of the nodes in a highly dense network. The approach adopted by in-network processing is essential for emerging applications such as sensor networks where resources such as bandwidth and energy are limited. In-networking query processing is critical for reducing network traffic when accessing and manipulating sensor data [3]. [18] discusses building of efficient wireless sensor networks where in-network aggregation techniques are applied to form a gradient between the different nodes in the system.

One of the most important areas where the advantages of sensor networks can be exploited is for tracking mobile targets. Scenarios where such network may be deployed can be both military (tracking enemy vehicles, detecting illegal border crossings) and civilian (tracking the movement of wild animals in wildlife preserves). Typically, for accuracy, two or more sensors are simultaneously required for tracking a single target, leading to coordination issues. Additionally, given the requirements to minimize the power consumption due to communication or other factors, we would like to select the bane essential number of sensors dedicated for the task while all other sensors should preferably be in the hibernation or off state. In order to simultaneously

satisfy the requirements like power saving and improving overall efficiency, we need large scale coordination and other management operations. These tasks become even more challenging when one considers the random mobility of the targets and the resulting need to coordinate the assignment of the sensors best suited for tracking the target as a function of time. In this paper we propose an algorithm for managing and coordinating a sensor network for tracking moving targets. The problem of tracking targets with sensor networks has received attention from various angles. In [8], Galstyan and et al. proposed a distributed online algorithm in which sensor nodes use geometric constraints induced by both radio connectivity and sensing to reduce the uncertainty of their position. In that algorithm, sensor nodes use online observation of a moving target to simultaneously improve both the moving target and their own positions. A small fraction of reference nodes are pre-planned or GPS placements into the network. In [17], Kirill and et al. presented a two-level cooperative tracking algorithm using binary-detection sensors to track the object with more precision and accuracy. In the first level phase, the local target position estimation is computed. Initially, the target is estimated to equal to the position of the sensor node. As more information from the other sensor nodes is available, the position estimate is recomputed as a weighted average of the sensor locations. Sensor nodes that lie closer to the path of the target receive more weight. These estimations are then aggregated to compute the path of the object, which produces a more precise estimate for the target location. In the second level, a piecewise linear approximation of the path is computed using a line-fitting algorithm on the positions obtained in the first level. In [7], Fang and et al. proposed a collaborative computation approach where they count the number of targets in sensing terrain. They equate the computation of the number of targets to the computation of leader nodes in the network with maximum signal strength. They aggregate this information to get a final report of the total count. The work described in [12, 15, 22] provides similar distributed collaborative algorithms for target localization, classification and tracking.

The balancing between computation and communication has been explained in a lot of work in in-network processing, this work is one of our motivations. The topographically addressed sensor nodes are similar to the way we have dealt with the placement of nodes in our work [13]. A distributed algorithm, which estimates the gradient of an environmental scalar variable such as temperature, intensity of light, atmospheric pressure, etc. using a random sensor network, is discussed by [23]. [1, 26] have examined ways to provide in-network aggregation for the internet in a traditional way. There is a concept of a router and a centralized node. But in the practical world of sensors and other similar devices, it is very expensive for a central node to communicate with all the other nodes in the system. Hence a hop by hop network is built where in each node can talk with its neighboring node [18]. In this aspect, the central idea of our paper is similar with [18]. There is no concept of a router or a centralized node in our work. This is the main difference from active networks; we assume that our algorithm would be primarily used for applications/networks where the bandwidth is limited and the communication costs between nodes are very high, but the computation costs at the local node is very less. In [9] algorithms have been designed which use a dense deployment of nodes to carry out in-network processing for data aggregation and similar problems. The main aim of this paper is similar to that of our paper. The algorithm proposed is used to compute data aggregations over a system of nodes by using in-network processing of the data stored in the nodes. We attempt to do the same, but there is a temporal attribute of data which is involved in our paper. One of the main differences with respect to [9] is that they do not attempt to address temporal attributes in data. Also, the methodology adopted is different in that they assume a two way flow of data between nodes. To locate sensors in an ad hoc network, we would need some kind of directory look up service. An algorithm is discussed by [10] that exploits the characteristics of ad hoc wireless sensor networks to discover position information of the sensor nodes even when they have been sprinkled all over the earth. Our work differs from this paper in that the sensor nodes do not have global knowledge of the topology of their physical location. But in our work, the nodes in the network need to know their global coordinates in order for them to start the in-network data analysis and aggregation. Jini [27] is another such example. It uses Java technology to provide directory service to the nodes in the system. Thus it is very well suited to a network with a lot of bandwidth. However our work is fundamentally different from these papers in that we do not support ad hoc networks. We assume that the nodes already have information about itself and the other relevant nodes. Hence there would be no exchange of information between nodes which is how we save on bandwidth. Ad hoc routing does not support in-network processing. In contrast to this the main aim of our paper is to determine global patterns in the network by using in-network aggregation.

Knowledge discovery and data mining are emerging fields, whose goals are to make sense out of large

amounts of data collected, by discovering hitherto unknown patterns. Many interesting and efficient data mining algorithms have been proposed [19, 20, 25]. These database-oriented mining algorithms can be classified into two categories: concept generalization-based discovery and discovery at the primitive concept level. Generalization of attribute values (or concepts) is the main idea in the former (one such example is the DB-Miner system) whereas the latter discovers strong regularities or association rules from databases without concept generalization. There hasn't been a lot of work in the area of mining temporal concepts. Most of the existing work is based on time series analysis of temporal sequences. [4] discusses some of the challenges posed by the temporal data. A few algorithms have been discussed which compute the temporal aggregation. There has been some work done by [18] in the field of parallel algorithms for temporal aggregation. In this paper, we discuss an algorithm which discovers temporal patterns among distributed databases.

The rest of the paper is organized as follows: In the following section, we give an introduction to the basic concepts. A step by step outline of our algorithm is described in section 4. We describe the trajectory of tracking in section 5. Section 6 analyzes the complexity of our algorithm. In section 6, we present our simulation results. In section 8 we discuss the practical matter. We conclude our work in section 9.

3 Problem Formulation Terms

We consider sensor nodes that spread in a grid across a geographical area and collaborate among themselves to establish a grid sensor network. Our development is in the context of temporal data being recorded by a number of sensors. We outline some of the ideas used in the development of our algorithm. The dataset used in our algorithm consists of X and Y coordinates of the points at which the light event is sensed, along with the timestamp. Each sensor may have a number of such data points recorded in its local memory. We refer to the term point to a combined set of information about moving object which includes x,y coordinates and the timestamp.

3.1 Local Hypothesis :(LH)

Forming of Local Hypotheses is the first step that is implemented in each of the sensor nodes. A Local Hypothesis is a set of three or more points, satisfying the following criteria

- Taking sets of three points or more, they should lie on the same line in the same direction;
- The points in the LH should be in ascending timestamp manner. The angle between two points p_1 and p_2 can be computed by the following equation:

$$Angle(p_1, p_2) = \arctan[(Ycord(p_2) - Ycord(p_1))/(Xcord(p_2) - Xcord(p_1))] \quad (1)$$

We define the points p_1, p_2, p_3 and p_4 to be lying in the same direction if the angle between any pair of points is the same as the angle between the other points e.g. in Figure 1, the points p_1, p_2, p_3 and p_4 represent a Local Hypothesis as is indicated by a rectangular box in the figure. This can be applied to points lying in a straight line in any direction. LH is considered as a straight line starts from the first point (FP), and ends with the last point (LP) with a specific angle of slop. For that we consider the structure of LH as the following: LH = (FP,LP, angle). In Figure 1, LH_1 has p_1 as first point (FP), LH_2 has p_9 as the last point (LP). In the same figure, p_{20} and p_{22} are at the same line but they are not form an LH because they are not passing through at least one more point.

3.2 Global Hypotheses

Global Hypothesis Component:(GH_c) A GH_c component is formed when different LHs at sensor node are merged in ascending manner according to their timestamps. The GH_c start at the first point in the first attached LH and ends at the last point in the last attached LH with a specific angle equal to the slop of any of merged LHs. We define the length of GH_c to be the number of LHs considered during forming GH_c. We define the GH_c structure as GH_c=*(Start Point (SP), End Point (EP), angle).*

Global Hypothesis:(GH) Forming GH is the second step of our algorithm. A GH is considered as a structure that contains set of GH_cs components. The GH starts at the start point in the first attached GH_c and ends at the end point in the last attached GH_c. We define the length of GH to be the number of GH_cs considered during forming GH. We keep track of GH direction by updating the *PointChangeList* which includes the points at which the GH changes its direction. We define the GH structure as *GH=(Start-Point-Of (GH), <PointChangeList>, End-Point-Of(GH))*. In Figure 1, the solid black line is the GH and the small boxes represent the points belong to *PointChangeList* at which the GH changes its direction.

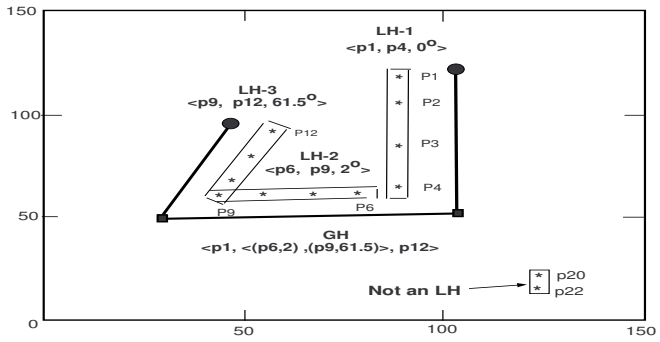

Figure 1: Example of LHs and GH

4 Algorithm Outline

There are two stages of our algorithm; the first stage will be executed at every sensor node in the system to generate the LHs (Local Computation Step), while the second stage will be executed at the temporary central nodes to generate the global hypothesis component GH_c (Global Computation Step). The main idea of this algorithm is to maximize the local processing of data at the local sensor node and minimize exchange of information, which produces meaningful results, between the nodes.

4.1 Algorithm Assumptions

The assumptions about the sensor network are the following:

1. All sensors have the same characteristics;
2. All sensors are spread in a grid like manner across the whole sensing area;
3. All sensors have the capability to capture and store the information of any moving object in their sensing range. The information includes the approximate x, y coordinates, and the timestamp about moving object. With different speeds of the moving object this may introduce local shifts into the trajectory i.e. the object trajectory follow similar paths, but certain sub-paths are shifted in time.

No specific assumptions are made about the movement pattern of the target. However we assume that the targets originate outside the sensing region and then move inside. Also, it is assumed that the aggregated data are reported to the end user.

4.2 Local Computations

Each node in the system collects all the required information about the moving target in its range, and form the LHs by executing the following procedure.

1. Output: Set of Local Hypotheses
2. Each node in the system, initially reads the data (say N points) from its database, and sorts it in ascending order according to their timestamps.

3. $k = 1$, $i = 1$ // k is the current LH number and $i = 1$ is the current point number
4. while($i < N$)
   (a) Find the angles $Angle1$ = angle(p_i, p_{i+1}) and $Angle2$ = angle(p_{i+1}, p_{i+2})
   (b) if ($Angle1 = Angle2$)
      i. LH_k.FP= p_i ; LH_k.LP=p_{i+2} ; LH_k .angle = $Angle2$.
      ii. for j =i+3 to N
         - if angle(p_{j-1}, p_j) = angle(p_{i+1}, p_{i+2})
           - $LH_k.LP = p_j$;
         - else
           - $i = j$, $k = k + 1$
           - Goto step 4
         - end if
      iii. end for
   (c) else $i = i + 1$
   (d) end if
5. end while
6. end Local Computations

Initially, each node arranges the recorded points in ascending manner according to their timestamps line 2. We take the first three points and find the angle between the first and the second point, and the angle between the second and the third point (line 4.(a)). In line 4.(b), if the computed angles in 4.(a) are the same, the LH is established by setting up the first point as the FP of the LH, the third point as the LP of the LH and LH angle will be the angle between the second and the third point. For further points may be added to the current LH, we take the next point p_j and determine its angle with the last added point to the current LH. If (p_{j-1}, p_j) angle equal to the LH angle, we update $LH.LP$ to p_j (line 4.ii). If the first three points are not at the same angle, we skip the first point of the three and start from the second one (line 4.c).

4.3 Global Computations

Every node in the system is associated with an index number which indicates the position of the node. If the index number is 1, then it denotes the node at the bottom left corner. Index 2 refers to its adjacent node on the right and the numbering of the index goes on in this way.

1. The list of LHs which is sent by the neighboring nodes is collected by the temporary central node and is added in one data structure which exists in the temporary central node. In this way, we can gather the details of the LHs of all the neighbors in the system.
2. Once the list of the LHs is collected, we go through them and merge them with the list of LHs of the temporary central node to form Global Hypothesis components for the temporary central node. The conditions which we check while merging these entries are whether the LHs of the different nodes lie in the same direction or not and whether the time stamps of the LHs in the nodes are in ascending order or not. The main criteria while forming the GH_cs is that the temporary central node should be a part of this GH_c.

4.3.1 Global Hypotheses at Zero Iteration

The following algorithm will be executed at every node in the system.

1. **Input:** A set of local hypotheses.
2. **Output:** Global Hypothesis components GH_cs

3. send your LHs to your neighbor nodes
4. wait to receive the set of LH_is from your neighbors
5. Add the received LHs to your DS_i
6. sort all LHs in DS_i according to their timestamps in ascending manner.
7. set $j = 1$ // j is the current number of GH_c
8. set $GH_{c(j)}.SP$=$LH_1.FP$, $GH_{c(j)}.angle$=$LH_1.angle$,
9. set Angle1 = angle(first LH in DS_i)
10. for every next LH_k in the arranged LHs of DS_i
    (a) Find Angle2 = angle(LH_k)
    (b) if (Angle1 = Angle2)
        i. Add LH_k to $GH_{c(j)}$
        ii. $GH_{c(j)}.EP$= $LH_k.LP$
    (c) else
        i. $j = j + 1$,
        ii. set Angle1 = Angle2
        iii. $GH_{c(j)}$=LH_k
        iv. $GH_{c(j)}.SP$=$LH_k.FP$
        v. $GH_{c(j)}.EP$= $LH_k.LP$
        vi. $GH_{c(j)}.angle$=$LH_k.angle$
11. end for
12. End Global Hypotheses at Zero Iteration

After the list of LHs is formed at each node in the system, every node will send its LHs to its neighbors. Then the node is ready to perform computations on the received lists of LHs to form the Global Hypothesis components, these lists of LHs are stored at data structure DS_i of the node N_i (lines 3-5). In line 6, the list of LHs that N_i has in its DS_i are arranged in ascending manner according to their timestamps. The components of global hypotheses (GH_cs) can be created as the following: if the object moves in one direction, all LHs are set into one GH_c (lines 10.a,b), but different GH_cs are created if the object changes its direction during the moving (lines 10.c).
After zeroth iteration, we have two options: we can either compute the entire global summarized pattern for all the nodes in the system or compute the entire global summarized pattern for a predefined *node* only.

4.3.2 First Approach: Entire Global Computations for all Nodes

In this approach, we compute the consolidated global hypothesis at every node N_i in the system. The GH_cs for every node is merged with the GH_cs obtained from the neighbors to obtain the consolidated GH.

1. **Input:** Set of global hypothesis components.
2. **Output:** The entire global hypotheses
3. Define *PointChangeList* as a list of elements, each element represents the location at which the object changes its direction. Each element is represented by the coordinates (x,y) of the location and the angle of changing, initialize *PointChangeList* to Φ
4. for (number of iterations required)
    (a) send your GH_cs to your neighbors

(b) wait to receive GH_cs from your neighbors
(c) Add all the GH_cs from the input list into your DS_i
(d) sort the GH_cs in ascending manner according to their timestamps
(e) Call MergeGHCs(GH_cs of DS_i) // merge the successive GH_cs that have the same angle and keep the results in DS_i
(f) Start-Point-of(GH)= $GH_{c(1)}.SP$
(g) End-Point-of(GH)= the end point of last GH_c
(h) The *PointChangeList* will be the start point ($GH_c.SP$) and the angle ($GH_c.angle$) of each GH_cs

5. end for
6. End Entire Global Computations for all the Nodes

At this point, every node in the system has its list of GH_cs that formed at zero iteration. In order to discover the actual pattern, we need to compute the consolidated global hypotheses for all the nodes in the system. First the number of iterations to be performed is determined by the user (line 4), where as the number of iterations increased more actual patterns discovered. We set all the GH_cs of the input list into the DS_i (line 4.c) then all the components are arranged according to their timestamps in ascending manner (line 4.d). As in pervious step the calculations are performed for all the nodes in the system, but in this step, we work with the GH_cs. For node N_i, we merge the GH_cs of N_i with its GH_cs neighbors by applying MergeGHCs procedure (lines 4.e). In lines 4.f,g,h, we get the global GH by finding the start point as the start point of the first GH_c, the end point as the end point of the last GH_c component, and the *PointChangeList* as the start point and the angle of each GH component in DS_i.

Procedure MergeGHCs(GH_cs of DS_i)

1. **Output**: Merge the successive GH_cs that have the same angle
2. set $i = 1$, // where i is the current number of GH_c in DS_i
3. set $j = 1$ // where j is the current number of new GH_c
4. set $GH_{c(j)}.angle = GH_{c(i)}.angle$, $GH_{c(j)}.SP = GH_{c(i)}.SP$, $GH_{c(j)}.EP$)= $GH_{c(i)}.EP$
5. angle1 = $GH_{c(i)}.angle$
6. for every next GH_c ($GH_{c(i+1)}$) in DS_i)
   (a) angle2 = $GH_{c(i+1)}.angle$
   (b) if (angle1 = angle2)
      - $GH_{c(j)}.EP = GH_{c(i+1)}.EP$
   (c) else
      - $j = j + 1$
      - angle1 = angle2
      - $GH_{c(j)}.angle = GH_{c(i+1)}.angle$
      - $GH_{c(j)}.SP = GH_{c(i+1)}.SP$
      - $GH_{c(j)}.EP) = GH_{c(i+1)}.EP$
   (d) end if
7. end for
8. End MergeGHCs procedure

The GH_cs merging procedure works as the following: In lines 4,5, we assign the start point, the angle, the end point of $GH_{c(1)}$ in the DS_i to the start point, the angle, the end point of new $GH_{c(j)}$. If the angle of the current GH component ($GH_{c(i+1)}$) is equal to the angle of the previous GH component ($GH_{c(i)}$), we merge $GH_{c(i+1)}$ to the $GH_{c(i)}$ otherwise, we initiate a new GH_{j+1} with the start point, the angle, the end point to the start point, the angle, and the end point of the $GH_{c(i+1)}$.

4.3.3 Second Approach: Entire Global Summarized Pattern for Target Node

Because of the high complexity of the first variant where we determine the GH for all the nodes in the system, we modify our requirements such that the global hypotheses for only a single node is required. First we introduce the concept of a *target node*.

Target Node target node is a predefined node in the system and it could be chosen as any node in the system but as a results of simulations it is advantageous to select the target node to be the middle node of the grid and that to reduce the number of exchanged messages.

Levels (Gradient) we define the gradient as the distance of each sensor node in the system to the target node.

$$Gradient(SensorNode) = Distancebetween(Targetnode, SensorNode)$$

This distance is calculated as the city block distance between the two nodes. We calculate the distance of all the nodes form the target node in terms of units.

$$Distancebetween(TargetNode, SensorNode) = CityBlockDistancebetween(Targetnode, SensorNode)$$

Once the gradient for all the nodes is calculated, we divide the nodes into levels. The level of the target node set to zero and that of its neighboring nodes is set to one, where the number of levels increase by increasing the distance from target node. The main idea behind assigning levels to the entire system of the sensor nodes is that can be only one way of communication between the nodes. This means that nodes at higher level only, can send messages to those at lower level and the nodes at lower are not allowed to send messages to the nodes at a higher level. In this way, we save a lot on communication cost.

Global Computations for Variant of Algorithm We introduce this step as a second variant of our algorithm. We assume that the summarized GH to be computed for a particular target node only which results in an increase in the efficiency of our algorithm. The target node is specified before we run our algorithm. According to the target node location, the level will be assigned for each node in the system by calculating the City Block distance of each node from the target node. Also we consider that the GH_cs at zero iteration already computed for every node in the system.
The second variant includes two procedures, the first procedure will be executed at the non target nodes and the second one will be executed at the target node

GH Computation Procedure for non Target Node N_i

1. Compute your distance from target node based on coordinates of the node
2. Assign level based on computed distance;
3. wait to receive list of GH_cs from neighboring nodes at higher level
4. sort the GH_cs in ascending manner according to their timestamps
5. Call MergeGHCs(GH_cs of DS_i)
6. Check the level of all the neighbors
7. Send the current GH so far to neighboring node at the lower level
8. End GH Computation Procedure for non Target Node

In lines 1,and 2, we compute the distance between the target node and the other nodes in the system and assign the levels to all the nodes in the system based on its distance from target node. The steps of exchanging of messages to form the summarized global pattern come after we assign the level for all the nodes. In lines 3,4, and 5 each node at the lower level receives the current GH_cs from its neighboring nodes at the higher

level, sorts the received GH_cs, and then merges the successive GH_cs that have the same angle and keep the results in DS_i by applying MergeGHCs procedure. In lines 6,7 each node checks all its neighbor levels, if they are at a lower level simply each node at the current level sends its merged GH_cs to its neighbors at the lower level.

The following procedure will be executed at the target node only

GH **Computation procedure at the Target Node**

1. **Output:** Global Hypothesis for target node
2. Assign zero as your level
3. wait to receive list of GH_cs from neighboring nodes at higher level
4. sort the GH_cs in ascending manner according to their timestamps
5. Call MergeGHCs(GH_cs of DS_i)
6. start-Point-of(GH) = $GH_{c(1)}.SP$
7. The *PointChangeList* will be the start point and the angle of each GH_cs in DS_i
8. End-Point-of(GH) = the EP of last GH_c
9. The consolidated GH that formed at the target node is the desired output
10. End GH Computation Procedure for target node

In line 2, we assign zero level for target node. In lines 3,4 and 5, nodes at the higher level send their GH_cs to the target node sorts the received GH_cs, and then combine the successive GH_cs that have the same angle by applying MergeGHCs procedure. In lines 6,7, and 8, the global GH start point will be the start point of the first GH_c, the end point will be the end point of the last GH_c component, and the *PointChangeList* will be the start point and the angle of each GH component in DS_i. the current GH will be the consolidated GH and the desired output.

4.3.4 The Two Variants Comparison

- The first variant computes the GHs of all the nodes in the system whereas the second variant will compute the GH for the target node only.
- In the first variant, there is no concept of a level whereas in the second variant, we use concept of levels extensively to obtain the end results.
- For the second variant, there is an initial step of computing the levels for each of the sensor nodes before the in-network aggregation can start. For the first variant, there is no such initial step.
- In the first variant, the information flow or message flow is in all possible directions whereas in the second variant, the information flow is from the outer level of the system towards the inner level (or towards the target node).
- The first variant requires more than one iteration to aggregate the data (or LHs) from the different sensor nodes to form the final result (GHs); the second variant requires only one iteration to perform the in-network aggregation and to get the final result. If the number of iterations is restricted, it is possible that the length of the GHs for the nodes in the system for the first variant is less than the would be final length of the GHs whereas in the second variant, the constraint on the number of iterations does not affect the length of the final GH.
- In the second variant, the nodes at the edge of the system do not form GHs of length more than three. In a similar way, we can say that the nodes just below those at the edge do not form GHs of length more than four. But in the first variant, any node in the system can form a GH of any length greater than or equal to three.

- In the first variant, the length of the GH increases by either one or two with every iteration depending on the placement of the node. In the second variant, this is not applicable as the final length is formed for the GHs for all the possible nodes at the end of the first iteration itself.
- The complexity of the first variant is much higher than the second variant. But the amount of information available in the first variant is also much more than the second variant.
- In the first variant, processing of the GHs for all the nodes is done in a parallel manner; the nodes compute the GHs independent of the other nodes. In the second variant, the processing of the GHs for the target node is done in a sequential manner. The processing of the GHs is first done for the nodes at a higher level and then is shifted to the nodes at the lower level.

5 Trajectory Description

The end user will have the summarized GH as a set of points ($< x$, y, $time >$ in 3-dimensional location-time) and the objective is to represent the trajectory of the moving object at the end user. A trajectory can be represented by a sequence of connected segments each of which joins two consecutive reported points. i.e. the start point of the reported summarized GH is connected by line segment to the first point in the *PointChangeList* and the first point in the *PointChangeList* is connected by line segment to the next point till the last point in the *PointChangeList* which is connected with the end point of the reported summarized GH. To produce these segments, one way is to use the interpolation schemes (Line Based Models) . These schemes create trajectories that have angles at reported locations which do not represent well the smooth trajectories of moving objects. The second used representation is the curve based trajectory representation model using Catmull-Rom spline which provides much more accurate trajectories than line-based models when we have the same amount of data. One of the features of using Catmull-Rom spline is that the specified curve will pass through all of the control points and this is not true of all types of splines.

In curve based trajectory representation model, we represent the trajectory by a sequence of curve segments, rather than line segments, each of which connects two consecutive points, where most natural moving objects, such as airplanes, vessels, and vehicles, draw a smooth trajectory with no angles. The parametric form of a third-order polynomial to obtain a spline is given by the following Equation.

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{2}$$

where a_0, a_1, a_2 and a_3 are constant coefficients. Theses coefficients are determined from several equations that reflect the properties of the cubic spline. To calculate a point on Catmull-Rom spline curve, two points on either side of the desired point are required. The point is specified by a value t that signifies the portion of the distance between the two nearest control points.

Given a GH of four points p_0 ,p_1, p_2, p_3. Catmull-Rom spline in equation form: (for a point q on the curve at t) will be

$$q(t) = 0.5 * ((-p_0 + 3 * p_1 - 3 * p_2 + p_3) * t^3 + (2 * p_0 - 5 * p_1 + 4 * p_2 - p_3) * t^2 + (-p_0 + p_2) * t + 2 * p_1) \tag{3}$$

t takes values between 0 and 1, and the curve passes through p_1 at $t = 0$ and p_2 at $t = 1$. To do more than two points, we can step through the array of points using the previous point, the current point, and the next two points as the four points for the spline. For each segment, we can draw a curve for $0 < t < 1$. This curve will be between the current point and the next point.

6 Complexity Computing

Since the effective lifetime of each sensor node is determined by its power supply, and since transmitting a single bit of data by sensor node is equivalent to 800 instruction execute at a sensor node (i.e. it is required a lot of energy to transmit a single message). For these reasons the most preferable system is the one which requires minimum number of message to be exchanged between the nodes as much as possible to preserve the amount of bandwidth and maximize the sensor node lifetime. Hence the main way to evaluate our system complexity will be in terms of the number of message exchanged between the nodes in the system. We

consider the two variants of our algorithm, However for both these variants, the initial two steps (computing of the local hypotheses and the global hypotheses at zero iteration) are common and hence we first calculate the complexity of these two steps.

6.1 Local Computation Cost

Each node in the system analyzes the collected data to extract the local patterns, since the extraction of the local patterns or forming the LHs occurs locally, therefore there is no communication cost.

6.2 Global Computation Cost

Complexity Computing of GH_cs at zero Iteration For computing the GH_cs at zero Iteration, each node gets the list of LHs from all of its neighbors, according to the placement of the node in the system the number of neighbors for any node ranges from three to eight. A node at the corner has three neighbors where as a node any where on the edge has five neighbors and a node any where else has eight neighbors.
If we consider $n \times n$ system in which we have n^2 nodes. The number of corners nodes is four, the number of nodes on the four edges is $4(n-2)$, and the number of nodes elsewhere in the system is $(n-2)(n-2) = (n-2)^2$. The total number of neighbors for all nodes in the system will be: $3*4+5*4(n-2)+8*(n-2)^2 = 8n^2-12n+4$. Therefore, the number of required messages for this step will be $8n^2 - 12n + 4$

6.2.1 Complexity Computing of entire GHs for all Nodes

In this step, we compute the entire GH for all the nodes in the system. Here again, the messages will be exchanged between nodes and all of their neighboring nodes, but the difference here is that these messages are to form a combined GH. The number of messages needed for this would be the same as before $8n^2 - 12n + 4$. After every iteration a new GH component ($GH_{resultant}$) added to GH or length of GH component increases by either one or more depend on their location in the system and every node replace its existing GH with the new GH which was formed during that iteration. The updated GH is used to exchanged with the neighboring nodes during the next iteration. In this way the messages keep getting exchanged between the nodes to eventually form a global summarized GH. If we require k iteration to form the complete GH, the total number of message exchanged would be $k * (8n^2 - 12n + 4)$. We define the best case and worst case computations for this system as follow.

Best Case: The least number of iterations to form the entire summarized global patterns is the best case. This occurs in and a round the center of the system, since every node in center have neighboring node on both sides, a new GH length added to their GHs during every iterations. Thus it takes only $n/2 - 1$ iterations to form the entire GH, then the complexity for this step will be: $(n/2 - 1) * (8n^2 - 12n + 4)$. where $k = (n/2 - 1)$, in the best case for the number of iterations for $n \times n$ system. Therefore the algorithm complexity in the best case will be:

$$Total\ Number\ of\ Exchanged\ Messages\ = (8n^2-12n+4)+(n/2-1)*(8n^2-12n+4) = 4n^3-6n^2+2n \quad (4)$$

Worst Case: Worst case is reverse to the best case where worst case is the case in which it takes the maximum number of iterations to form the entire summarized global patterns. This occurs for the nodes on the edge and at the corner of the system. Since they have neighboring node on only one side to contribute to the growth of their GHs during every iteration the length of their GH increases by only one and thus it takes $n - 1$ iteration to form the entire GH, therefore the complexity will be $(n - 1)(8n^2 - 12n + 4)$. Therefore the algorithm complexity in the worst case will be:

$$Total\ Number\ of\ Exchanged\ Messages\ = (8n^2-12n+4)+(n-1)*(8n^2-12n+4) = 8n^3-12n^2+4n \quad (5)$$

6.2.2 Complexity Computing of GH for Target Node only

In this case the user specifies a predefined target node from among the nodes in the system. The GH are then computed for this node only.
The first step in this case is that the target node informs all the nodes in the system about itself being the

target node. This is done through exchange of messages, which requires $8n^2 - 12n + 4$ (in the worst case) messages. Which is the total number of neighbors for all the nodes in the system.
The next step is the forming of the levels in the system based on the gradient of each node, where in this step starting from level zero or the target node, levels are assigning to all the nodes in the system in sequential manner. The number of messages required to do this is $8n^2 - 12n + 4$.
Once all the level are assigned, messages start getting exchanged from nodes in the outer level to those in the inner level. If we assume the average number of levels to be assigned is L, average number of nodes in each level is n_0, and the average number of neighboring nodes in the lower level for any node in the higher level is n_i then the total number of exchanged messages is $L * n_0 * n_i$. Therefore the total number of message required to compute the GH for the target node is given by

$$Total\ Number\ of\ Exchanged\ Messages = 2(8n^2 + 12n + 4) + L * n_0 * n_i \quad (6)$$

From equations 5, and 6 we can conclude that the complexity of computing the consolidate GH is reduced from $O(n^3)$ to $O(n^2)$ in the case of second approach (target node only).

7 The Simulation Results

The algorithms were programmed and tested by Java language. In the initial tests, 25, 36, 49, 64, 81, and 100 nodes were placed in a grid like manner ((5 × 5), to (10 × 10)).

We choose the following issues to generate results to compare the two variants:

- The total number of nodes in the system,
- The location of the target node in the system (This will be applicable to second approach only),
- The total number of iterations (This will be applicable to first approach only), and
- The total number of exchanged messages.

Here we present the results of the tests we had run for both variants of our algorithm.

- **The total number of nodes in system:** In the first approach, the number of messages exchanged between nodes in the system increase with the increasing of the grid size. The increase in size of the grid from 5 × 5 to 10 × 10 implies increasing the number of messages by almost 66%.

  In the second approach, we fixed the target node and varied the size of the grid from 5 × 5 to 10 × 10, this implies increasing the number of messages by almost 50%.

- **The location of target node in the system:** We compare the number of messages exchanged with different positioning of the target node in the system. We assumed that the size of the grid is 10 × 10 with the total number of nodes in the system to be 100. We vary the position of the target node from the center of the grid to the edge of the grid and to the corner of the grid. The number of levels formed is lesser when the target node is placed at the center than at the edges or at the corner. We found that the number of messages increases with the number of levels in the system. Hence the complexity or the number of messages increases when the target node is placed at the edge or at the corner of the system.

  Figure 2 shows comparison between the numbers of messages exchanged and the location of the target node. As expected the number of messages when the target node at center is less than the number of messages when the target node at the edge or the corner.

- **The total number of iterations:** We refer to an iteration as a procedure in which the nodes receive GH_cs from their neighboring nodes and merge them with their own GH_cs based on the criteria to form a consolidated GH. The length of GH increases with every iteration. We found that we require more than one iteration to obtain the entire list of summarized GHs. In our implementation we did vary the number of iterations from one to eight, we found the following:
  When we increase the number of iterations, the number of messages, and the amount of information

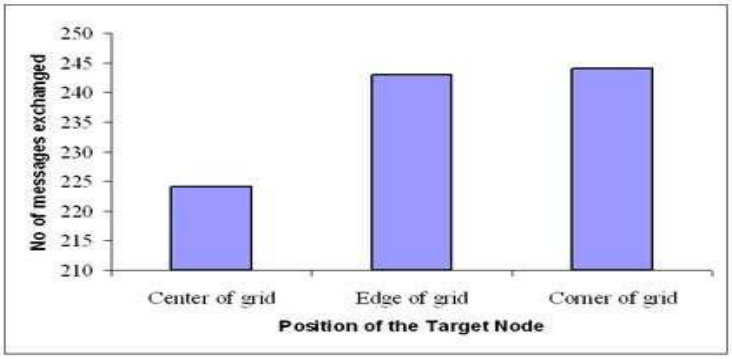

Figure 2: The number of Exchanged Messages versus the target node location

will be increased.
With every iteration the number of nodes, that form a part of GH, increases. We assumed that the node in consideration is a node on the edge or at the corner.
In a more generic way, we can say that the number of iterations required for a node on the edges or on the corner is equal to the grid size and the nodes elsewhere require a total of (grid size/2 - 1) iterations. Also, the number of neighbors for any node is dependent on the position of the node. Since a node on the edge or the corner has a lesser number of neighboring nodes than any other node in the system, and then it takes a higher number of iterations to compute its entire GH.

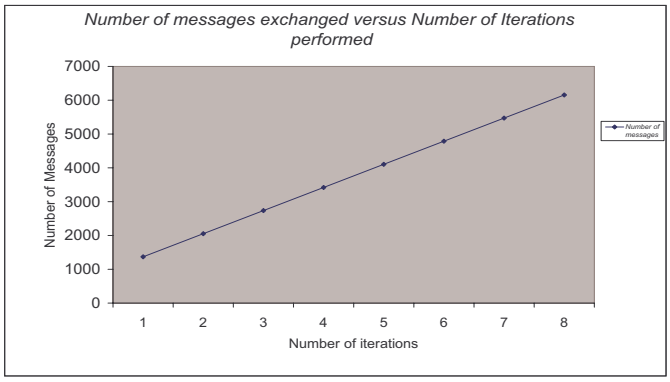

Figure 3: Exchanged messages versus the number of Iterations - for a node not on edges/corner

The nodes at the corner or at the edge need about double the number of iterations to build their entire GH when compared to the other nodes in the system, see Figure 3.

- **The total number of exchanged messages:** We found that using the second variant the number of messages is reduced. If we used 10*10 system of sensor nodes we found that:

  1. We require 684 exchanged messages to form the GH_cs at each node in the system.
  2. For the first variant, we require 6165 exchanged messages for all the nodes in the system to form the global entire summarized pattern,

3. For the second variant, we require 1611 exchanged messages to form the global entire summarized pattern at target node. The explanation of these messages will be as the following:
   (a) 684 exchanged messages to inform all the nodes in the system about the target node.
   (b) 684 exchanged messages to assign all the levels to all the nodes in the system based on the gradient of each node.
   (c) Once all levels are assigned, as shown in Figure 4 (there are nine levels are assigned and the target node at the corner of the system). In levels nine,eight, seven, and six we have 19, 17, 15, and 13 nodes respectively will exchange 51, 45, 39,and 33 messages to the neighboring nodes in the inner level. In levels five, four, three, and two we have 11, 9, 7, and 5 nodes respectively will exchange 27, 21, 15, and 9 messages to its neighbor nodes in the inner level, and for level one only 3 nodes will exchange 3 messages to the target node.

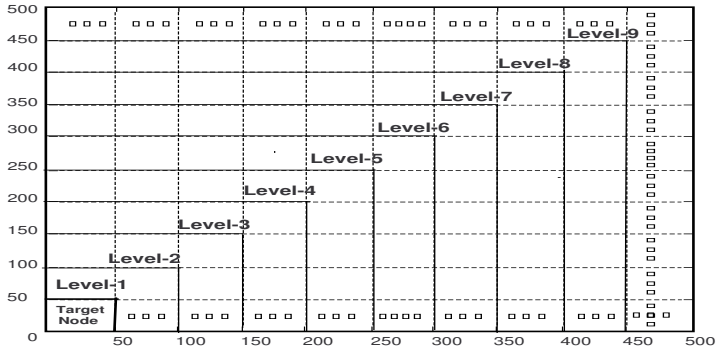

Figure 4: Target node at the corner in 10x10 system of sensor nodes

Hence we can see that in first variant there is 6156 message exchanged in the network to discover global spatiotemporal pattern but in the second variant required only 1611 message, therefore it is clear that there is significant amount of saving in messages between the two variants, see Figure 5.

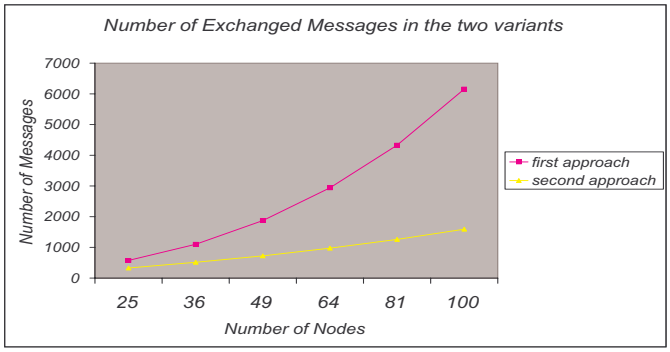

Figure 5: Exchanged Messages in the two Variants

8 Practical Matters Comments

Node Failure We discuss the recovery scheme in the presence of link/node failure, our recovery schema is very simple to be suitable for sensor networks. The recovery process comes as follows: The base station starts

to periodically send *connect messages*. When a sensor receives a connected message, it reply to base station by *connect messages* to inform the base station about its existence. Connect messages that are periodically sent by sensor node or base station are very small, each message consists of approximately 1 bit of data (1 indicate the existence of alive node else indicate that the death of the node). Whenever the time out of the base station out, and there is no response back from the sensor node the base station know this node needs to be replaced.

Message Loss/Collision are very common in application involving wireless communication because of the frequent environmental disturbances in the channel. Such situations are very difficult to detect at the sending sensor as it is not sure whether the message has reached its neighboring or intended sensor. Thus, to enable detection of messages losses at sending sensor, an acknowledgment signal can be used from the neighboring sensor to indicate proper reception of the message. If the sensor dose not receives such an acknowledgement signal after a time limit, then it can retransmit the message to enable proper working of the algorithm.

Communication Reliability In our setting we assumed that there was communication reliability in the network. There was some work focuses on the problem of communication reliability in sensor network, example of this work is in [21].

Localization Many sensor network applications require that the locations of the individual sensors be known, since sensor readings are in general of little use without geographic context. However, the same attributes that make sensor networks attractive make obtaining this information difficult. By placing a large number of relatively cheap sensors, it is possible to obtain many accurate measurements from sensors close to phenomena of interest; however, the sheer number of sensors and the need to minimize costs precludes manually recording the sensors' locations. It also precludes brute force solutions such as equipping each sensor with a GPS unit. Consequently, we would like the sensors to determine their own positions after placement. This is known as localization, and is typically achieved by having each sensor compute range measurements to its neighboring sensors, then algorithmically embedding the graph formed by these ranges into a coordinate system. This coordinate system is then used to perform location-dependent tasks such as geographic packet routing or target tracking. Localization is often complicated by the difficulty of obtaining enough accurate pair-wise range measurements between sensors. Inter-sensor ranges can be corrupted by noise or lost entirely due to occluded line-of-sight. Thus, consistently accurate localization requires robustness in the face of missing or low quality measurements. Nevertheless, localization is rarely if ever the purpose of a network. Sensor networks are typically deployed to observe active phenomena in the environment, and require accurate localization as a means to that end. As a result, there is pressure in localization research to achieve accuracy and robustness using as little hardware as possible. Target tracking is one of the motivating localization-dependent applications of sensor networks. In tracking applications, sensors jointly observe phenomena, which may be people or objects passing through the network or physical effects such as bullet shock-waves or anomalous sounds. Once a phenomenon is detected, the sensors collaborate to determine its spatial location. This estimate is reported to a computer or person monitoring the network. In our algorithm we consider that each sensor node is aware of its own location. Where given the locations of the sensors and accurate range information to the target, it is straightforward to determine the target's position based on the discovered sensor positions. Consequently, a localization error in our case depends on the specialized localized algorithm which handle localization problem.

Calibration In an ideal world, sensors would arrive from the factory fully calibrated to begin taking accurate measurements of their surroundings. However, this ideal situation is rarely achieved. For instance, deployment conditions such as temperature affect the accuracy of ranging algorithms based on acoustic time-of-flight by altering the speed of sound. Furthermore, as shown in [28], differences between sensors can also result in mis-calibrations that are difficult to correct before deployment. Calibration in the field can therefore offer meaningful improvement in both localization and target tracking accuracy. As with localization, there is considerable economic incentive to develop auto-calibration algorithms that allow sensors to self-calibrate in the field without external intervention.

9 Conclusions

In our work, we considered the problem of mining temporal data in distributed datasets. We worked with sensor nodes which were capable of capturing and storing approximate coordinate information about a

moving object or a target object. These nodes were placed in a grid and our algorithm was used to predict the nonlinear trajectory of the moving object. We considered this equivalent to mining of global spatiotemporal patterns from geographically distributed datasets. Since there is only one database scanning required in the beginning, the complexity of our algorithm reduced. The concept of maximizing the computations at the local sites and minimizing the exchange of messages between nodes help reduce the load on the network. This formed the crux of our algorithm. We reduced the complexity further, by introducing a variant of our algorithm wherein the global patterns are required for a single node only. We defined a target node and levels in the system. Then we went on to show that the number of exchange of messages required in the second variant of our algorithm was much lesser and that the complexity of the second variant falls down to as low as 33% of the first variant of our algorithm. We then did a compare and contrast of the two variants and presented results obtained for the various test cases for both the variants of our algorithm.

References

[1] Amir E., McCanne S. and Katz R. H., *An active service framework and its application to real-time multimedia transcoding*, In Proceedings of the ACM SIGCOMM conference; pages 178-189, Vancouver, Canada, 1998.

[2] Bhardwaj M., Garnett T., and Chandrakasan A. P., *Upper bounds on the lifetime of sensor networks*, Proceedings of IEEE ICC'01, Helsinki, Finland, 2001, p.785.

[3] Bonfils J.B. and Bonnet P., *Adaptive and Decentralized Operator Placement for In-Network Query Processing*, In Proceedings of the Second International Workshop on Information Processing in Sensor Networks; pages 47-62, Palo Alto, CA, 2003.

[4] Bongki Moon, Ines Fernando Vega Lopez, and Vijaykumar Immanuel., *Scalable algorithms for large-scale temporal aggregation*, Technical Report TR 98-11, Tucson, AZ 85721, 1998.

[5] Broch J., Maltz D.A., Johnson D.B., Hu Y.C. and Jetcheva J., *A performance comparison of multi-hop wireless ad hoc network routing protocols*, In Proceedings of the ACM/IEEE International Conf. on Mobile Computing and Networking, Dallas, TX, 1998, p. 85.

[6] Chang J. H. and Tassiulas L., *Energy conserving routing in wireless ad hoc networks*, Proceedings of IEEE INFOCOM'00, Tel Aviv, Israel, 2000, p.22.

[7] Qing Fang, Feng Zhao and Leonidas Guibas, *Counting Targets: Building and Managing Aggregates in Wireless Sensor Networks*, Palo Alto Research Center (PARC) Technical Report, 2002.

[8] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman, and Sundeep Pattern, *Distributed Online Localization in Sensor Networks Using a Moving Target*, IPSN'04 , 2004.

[9] Heidemann J., Silva F., Intanagonwiwat C., Govindan R., Estrin D. and Ganesan D., *Building Efficient Wireless Sensor Networks with Low Level Naming*, In Proceed. of the ACM Symposium on Operating Systems Principles, Chateau Lake Louise, Canada, 2001, p.146.

[10] Heinzelman W.R., Chandrakasan A. and Balakrishnan H., *Energy-efficient communication protocols for wireless microsensor networks*, In Proceedings of the Hawaii International Conference on Systems Sciences, Wailea Maui, HI, 2000, p.4.

[11] Intanagonwiwat C., R. Govindan, and D. Estrin., *Directed diffusion: a scalable and robust communication paradigm for sensor networks*, Proceedings of ACM MobiCom'00. Boston, MA, USA, 2000, p.56.

[12] Aslem Javed , Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus, *Tracking a Moving Object with a Binary Sensor Network*, ACM Sensys 03 , 2003.

[13] Khedr Ahmed M, Raj Bhatnagar *A Decomposable Algorithm for Minimum Spanning Tree* Distributed Computing- Lecture Notes in Computer Science Springer-Verlag Heidelberg Volume 2918 / 2004 pp. 33-44

[14] Lindsey S. and Raghavendra C.S., *Energy efficient broadcasting for situation awareness in ad hoc networks*, Proceedings of ICPP'01, Valencia, Spain, 2001, p.149.

[15] Liu J.J., Liu J., Reich J., Cheung P., and Zhao F., *Distributed group management for track initiation and maintenance in target localization applications*, In Proc. of 2nd workshop on Information Processing in Sensor Networks (IPSN), 2003.

[16] Manjeshwar A. and Agrawal D.P.., *TEEN: A routing protocol for enhanced efficiency in wireless sensor networks*, Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing. San Francisco, CA, 2001, p.2009.

[17] Kirill Mechitov and Sameer Sundresh ,*Cooperative Tracking with Binary-Detection Sensor Networks*, ACM Sensys' 03, 2003 .

[18] Nagpal R., Shrobe H. and Bachrach J., *Organizing a global coordinate system from local information on an amorphous computer*, Technical Report AI Memo No. 1666, MIT Artificial Intelligence Laboratory, 1999.

[19] Rajeev Rastogi and Kyuseok Shim., *Mining Optimized Association Rules with Categorical and Numeric Attributes*, IEEE Transactions on Knowledge and Data Engineering, 14(2002), p.29.

[20] Rakesh Agrawal, John C. Shafer, *Parallel Mining of Association Rules*, In IEEE Trans on Knowledge and Data Engineering, 1996.

[21] Senug Jong Park , Raghupathy Sivakumar, *Poster: Sink to Sensor Reliability in Sensor Networks*, MobiHoc'03, Annapolis, Maryland, USA, 2003.

[22] Jaewon Shin, Lenidas Guibas and Feng Zhao, *A distributed Algorithm for Managing Multi-target Identities in wireless Ad hoc Sensor Networks*, 2nd workshop on Information Processing in sensor network (IPSB'03), Palo Alto, California , 2003.

[23] Slobodan S.N. and Shankar S., *Distributed Gradient Estimation Using Random Sensor Networks*, In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, 2002.

[24] Sohrabi K., Gao J., Ailawadhi V., and Pottie G. J., *Protocols for self-organization of a wireless sensor network*, IEEE Personal Communication, 2000, p.16.

[25] Ramakrishnan Srikant, Quoc Vu, Rakesh Agrawal. *Mining Association Rules with Item Constraints*. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, August 1997.

[26] Tennenhouse D.L., Smith J.M., Sincoskie W.D., Wetherall D.J. and Minden G.J., *A Survey of Active Network Research*, IEEE Communications Magazine; pages 80-86, 1997.

[27] Waldo J., *The Jinni architecture for network-centric computing*, Communications of the ACM, 1999, p.76.

[28] Whitehouse C., *The design of calamari: an ad hoc localization system for sensor networks*, Master's thesis, University of California at Berkeley, 2002.