Computing and Informatics, Vol. 27, 2008, 853–873

A NEW MECHANISM FOR TRACKING A MOBILE TARGET USING GRID SENSOR NETWORKS

Ahmed M. Khedr

Mathematics Department, Faculty of Science
Zagazig University, Egypt
e-mail: amkhedr@zu.edu.eg

Manuscript received 9 May 2006; revised 2 July 2008
Communicated by Ladislav Hluchý

Abstract. Tracking moving targets is one of the important problems of wireless sensor networks. We have considered a sensor network where numerous sensor nodes are spread in a grid like manner. These sensor nodes are capable of storing data and thus act as a separate datasets. 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: Target tracking, sensor networks, in-network aggregation, spatio-temporal mining

Mathematics Subject Classification 2000: MS2000 68T05

1 INTRODUCTION

A sensor network consists of a set of 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. Sensor networks promise novel applications in several domains, for example forest fire detection, battlefield surveillance, or monitoring of human physiological. 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. Thus, energy conservation is of prime consideration in sensor network algorithms in order to maximize the network's operational lifetime.

In this paper, we assume that stand alone sensors are spread in a grid like manner in a region to be supervised. They are capable of sensing predefined parameters such as temperature, motion and geographical coordinates of objects in their vicinity and it is of interest to determine global patterns in a geographically distributed collections of 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.

We first characterize a particular type of tracking problem in region populated by sensor nodes that have limited wireless communication capabilities. We introduce a new distributed mechanism to perform data mining on the data stored in sensor nodes to extract useful information then a further round of exchange of messages is done to discover global spatio-temporal patterns from these sensors which represent the track of the moving target. our mechanism can be readily extended for multitargets as we discuss in the end of this work.

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. 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 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. The topographically addressed sensor nodes are similar to the way we have dealt with the placement of nodes in our work [11, 12].

The rest of the paper is organized as follows: In the next section, we discuss the related work. In Section 3, 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 7, we present our simulation results. We conclude our work in Section 8.

2 RELATED RESEARCH

Surveillance and monitoring applications using sensor networks have attracted considerable attention in the research community during the past few years. These kinds of applications form a canonical class of applications which can be constructed with sensor networks. The work presented in this paper is based on and inspired by various other research efforts. In the context of this paper, the prior work can be primarily classified into two categories. One relates to the different ways of constructing aggregation routing structures and the other relates to computing various global quantities using aggregation.

Target tracking with sensor networks was extensively studied by researchers from various areas. Different approaches were suggested using signal processing techniques, and aggregation techniques. The work presented in [5] uses collaborative signal processing of wideband acoustic and seismic signals for source localization and beamforming in an energy-constrained sensor network. In [13], the authors discuss similar collaborative signal processing techniques to detect, classify and track multiple targets. The authors assume that multiple target detection at each sensor is separated either in space or time, i.e. two targets are either separated by some distance or appear in two different time durations. In [2], the authors proposed a collaborative computation approach where they count the number of targets in a 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 [6, 14, 18, 20, 21, 22] provides similar distributed collaborative algorithms for target localization, classification and tracking. The work presented in this paper is similar to the goals addressed in these papers, but the approach suggested is at the application level involving collaborative computation rather than at the signal processing level.

In the context of information dissemination and data fusion in sensor networks, in-network aggregation has attracted considerable attention due to the data-centric nature of computations in sensor networks. In [10], the authors presented a novel approach called Directed Diffusion which addressed the data dissemination and aggregation tree building in sensor networks. This novel protocol addressed the data-centric dissemination, reinforcement based adaptation to empirically best path and in-network data aggregation and caching. The work in [3] discussed the impact of data aggregation on computation in sensor networks. The authors clearly discussed the advantages of using data centric approach in sensor networks as opposed to address centric approach used in traditional networks. The authors also examined the impact of source-destination placement and communication network density on the energy costs and delay associated with data aggregation. In [17], Madden et al. presented a TAG (Tiny Aggregation) approach for aggregation in low power, wireless sensor network environments. TAG allows users to express simple, distributed queries and have them efficiently executed in sensor network using in-network aggregation. The authors also discuss the properties of aggregates and show how these properties affect the performance of in-network aggregation. The papers [8, 16, 25] give details on in-network aggregation, aggregate representation and computation mechanisms using aggregation trees. The work presented in this paper is closest to the work described in the above papers, but differs from them in that a temporal attribute of data is involved in our work. In contrast to this the main aim of our paper is to determine global patterns in the network by using in-network aggregation.

In [24], a clustering based approach is proposed in which sensor nodes perform sense-predict-communicate-sense tasks as required by the cluster heads. In another approach [7], sensors detect the presence of a target for a threshold value. Nodes broadcast an alert message when this threshold value is reached and three similar messages have been received from their neighbor nodes. A trajectory of moving target is estimated as nodes alert their neighbor nodes while broadcasting these messages. In [15], the authors presented a grid based approach to address the problem of continuous delivery of data from the source to the mobile sinks.

In [19], Kirill 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.

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 for example [1, 23]. 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. In this paper, we discuss an algorithm which discovers temporal patterns among distributed datasets.

3 PROBLEM FORMULATION TERMS

We consider sensor nodes that spread in 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 terms 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 [9]. We refer to the term point to a combined set of information about moving object which includes x, y coordinates and the timestamp. Each sensor may have a number of such data points recorded in its local memory.

3.1 Local Hypothesis

A Local Hypothesis (LH) is a set of three or more points, satisfying the following criteria:

1. Taking sets of three points or more, they should lie on the same line in the same direction;
2. 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:

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

Figure 1 shows that the points p_1, p_2, p_3, and p_4 make a local hypothesis as indicated by a rectangular box. LH is considered as a straight line starting at the first point (FP), and ending with the last point (LP) with a specific angle. For that, we consider the structure of LH as (FP, LP, angle).

3.2 Global Hypothesis

A Global Hypothesis (GH) is formed when different LHs are merged in ascending manner according to their timestamps. We keep track of GH direction by updating the *PointChangeList* which includes the points at which the GH changes its direction. The GH starts at the first point of the first attached LH and ends at the last point of the last attached LH. We define the GH structure as (Start Point (SP), ⟨*PointChangeList*⟩, End Point (EP)). We define the length of GH to be the number of LHs considered during forming GH. In Figure 1, the solid black line is the GH, and the small boxes represent the *PointChangeList*.

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), while the

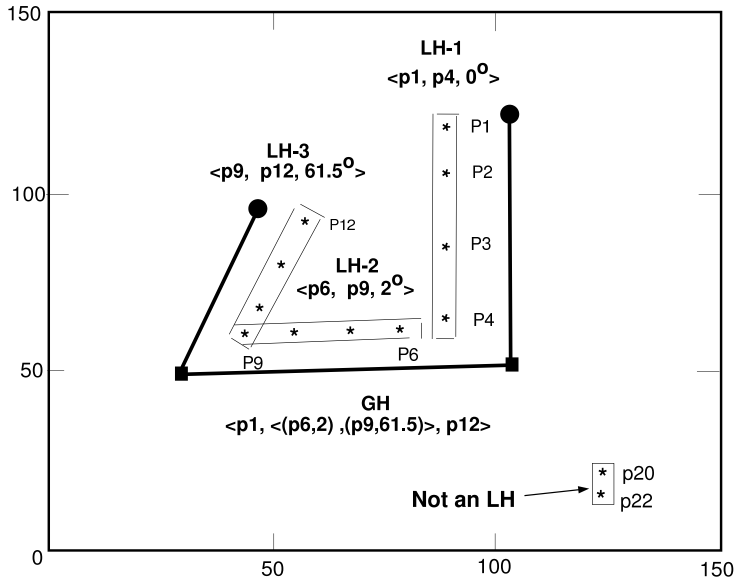

Fig. 1. Example of LHs and GH

second stage will be executed at the temporary central nodes to generate the global hypothesis component GH_c (Global Computation). The main goal of this algorithm is to maximize the local processing of data at the local sensor nodes and to minimize exchange of information, which produces meaningful results, between the nodes.

Algorithm Assumptions
The assumptions about our sensor network are as follows:

1. All sensors have the same characteristics;
2. All sensors have the capability to capture the information of any moving object in their sensing range. The information includes the approximate x, y coordinates, and the timestamp.

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, the aggregated data are reported to the end user. The outline of the whole mechanism is as follows.

4.1 Step 1: Local Computations

Every sensor node that has recorded points will execute the following steps to generate LHs:

- Initially, each node arranges the recorded points in ascending manner according to their timestamps.
- Compute the angles between the first and the second points, and between the second and the third points.
- If the computed angles are not equal, we skip the first point of the three points, and start from the second one; otherwise the LH is established by setting up the first point as the FP, the third point as the LP 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 equals to the LH angle, we update $LH.LP$ to p_j, otherwise we create a new LH starting from p_j.

The following is the Local Computation Pseudocode:

```
1.  sort your data in ascending order according to their timestamps
2.  let $i = 1$
3.  while ($i < N$), where $N$ is the number of points in the database
4.      compute the angles between $p_i$, $p_{i+1}$, and $p_{i+1}$, $p_{i+2}$
5.      if the angles between the three points are the same
6.          establish $LH$ by setting up the first point as $FP$ and the third point as
            $LP$ of the $LH$
7.          for every next point $p_j$ ($j = i + 3$ to $N$)
8.              compute the angle between $p_j$ and last added point to the current $LH$
                ($p_{j-1}$)
9.              if the computed angle equals to the previous one in step 4
10.                 add the point $p_j$ to the current $LH$ and update $LP$ to $p_j$
11.             else start creating a new $LH$ by setting $i = j$ and go to step (3)
12.         end for
13.     else $i = i + 1$ and go to step (3)
14. end while
15. End Procedure.
```

4.2 Step 2: 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 criterion while forming the GH_cs is that the temporary central node should be a part of this GH_c.

4.2.1 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, and then the node is ready to perform computations on the received lists of LHs to form the Global Hypothesis components, by executing the following steps:

- Store the received LHs in your data structure and arrange them in ascending manner according to their timestamps.
- The components of global hypotheses (GH_cs) can be created as follows:
  - if the object moves in one direction, all LHs are set into one GH_c,
  - different GH_cs will be created if the object changes its direction during the moving.

The following is the Pseudocode of creating Global Hypothesis components:

```
1. send your $LH$s to your neighbors
2. wait to receive the set of $LH_i$s from your neighbors
3. add the received $LH$s to your $DS_i$
4. sort all $LH$s in $DS_i$ according to their timestamps in ascending manner
5. set $j = 1$,   where $j$ is the current number of $GH_c$
6. set $GH_{c(j)}.SP$=$LH_1.FP$, $GH_{c(j)}.angle = LH_1.angle$, Angle1 = $GH_{c(j)}.angle$
7. for every next $LH_k$ in the arranged $LH$s of $DS_i$
   a) set Angle2 = $LH_k.angle$
   b) if (Angle1 = Angle2)
      i. add $LH_k$ to $GH_{c(j)}$, and set $GH_{c(j)}.EP$= $LH_k.LP$
   c) else
      i. $j = j + 1$,
        ii. set Angle1 = Angle2, $GH_{c(j)}.SP = LH_k.FP$, and
        iii. $GH_{c(j)}.EP = LH_k.LP$, and $GH_{c(j)}.angle = LH_k.angle$
8. end for
9. End Global Hypotheses at Zero Iteration.
```

After executing zero iteration, we have two options: The first one is to compute the entire global summarized pattern for all the nodes in the system, or to compute the entire global summarized pattern for a predefined *node* only.

4.2.2 First Approach: Computing Global Hypothesis at All Nodes

In this approach, we compute the consolidated global hypothesis at every node N_i in the system from the set of global hypothesis components. The GH_cs for every node is merged with the GH_cs obtained from the neighbors to obtain the consolidated GH. This can be done by doing the following steps:

1. First the number of iterations to be performed is determined by the user, where as the number of iterations increased more actual patterns have been discovered.
2. Store all the GH_cs of the input list into your data structure; then arrange the components according to their timestamps in ascending manner.
3. Merge your GH_cs with the GH_cs of your neighbors by calling **MergeGHCs** procedure.
4. Create the global GH structure by setting the start point as the start point of the first GH_c, the end point as the end point of the last attached GH_c component, and the *PointChangeList* will be returned from **MergeGHCs**.

The following is the Pseudocode of creating Computing global hypotheses at all nodes:

```
1. for (number of iterations required)
    a) send your $GH_c$s to your neighbors
    b) wait to receive $GH_c$s from your neighbors
    c) add all the $GH_c$s from the input list into your $DS_i$
    d) sort the $GH_c$s in ascending manner according to their timestamps
    e) call MergeGHCs($GH_c$s of $DS_i$) // merge the successive $GH_c$s 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 attached $GH_c$
2. end for
3. End Entire Global Computations for all the Nodes
```

Merging Global Hypotheses Components

The GH_cs merging procedure works as follows: assign the start point, the angle, the end point of first component in your data structure to the start point, the angle, the end point of new $GH_{c(j)}$, respectively. If the angle of the current GH component ($GH_{c(i+1)}$) is equal to the angle of the previous GH component ($GH_{c(i+1)}$), 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, and the end point to the start point, the angle, and the end point of the $GH_{c(i+1)}$. In this case, the start point and the angle of the new GH_{j+1} will be added to the *PointChangeList*.

Procedure MergeGHCs(GH_cs of DS_i)

```
1. set $PointChangeList = \Phi$
2. set $i = 1$, $j = 1$, where $i$ is the current number of $GH_c$ in $DS_i$ and $j$ is the
   current number of new $GH_c$
3. set $GH_{c(j)}.angle = GH_{c(i)}.angle$, and
4. $GH_{c(j)}.SP = GH_{c(i)}.SP$, $GH_{c(j)}.EP = GH_{c(i)}.EP$
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
        • $j = j + 1$
        • set Angle1 = Angle2
        • $GH_{c(j)}.angle = GH_{c(i+1)}.angle$
        • $GH_{c(j)}.SP = GH_{c(i+1)}.SP$, $GH_{c(j)}.EP = GH_{c(i+1)}.EP$
        • add the point ($GH_{c(j)}.SP$, $GH_{c(j)}.angle$) to the $PointChangeList$
    d) end if
7. end for
8. return ($PointChangeList$)
9. End MergeGHCs procedure.
```

4.2.3 Second Approach: Computing Global Hypothesis at a 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 following concepts:

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 our 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) is defined as the distance of each sensor node in the system to the target node.

$$Gradient(Sensor\ Node) = Distance\ between(Target\ node, Sensor\ Node) \quad (2)$$

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

We introduce the second approach of global hypotheis as 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

1. compute your distance from target node based on coordinates of the node
2. assign your 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. send the current GH so far to neighboring node at the lower level.

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 the 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 by calling MergeGHCs and keep the results in DS_i. In line 6 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 lower level, the *PointChangeList* will be returned from MergeGHCs.

GH Computation Procedure at the Target Node

1. assign zero as your level

```
2. wait to receive list of $GH_c$s from neighboring nodes at higher level
3. sort the $GH_c$s in ascending manner according to their timestamps
4. call **MergeGHCs**($GH_c$s of $DS_i$)
5. set start-Point-of($GH$) = $GH_{c(1)}.SP$, End-Point-of($GH$) = the $EP$ of last $GH_c$.
```

In line 1, we assign zero level for target node. In lines 2, 3 and 4, nodes at the higher level send their GH_cs to the target node which sorts the received GH_cs, and then combine the successive GH_cs that have the same angle by applying **MergeGHCs** procedure. In line 5, 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.

4.2.4 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 that 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 that of the second variant; but the amount of information available in the first variant is also much higher than with the second variant.
- In the first variant, processing of the *GH*s for all the nodes is done in a parallel manner; the nodes compute the *GH*s independent of the other nodes. In the second variant, the processing of the *GH*s for the target node is done in a sequential manner. The processing of the *GH*s 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 ($\langle x,\ y,\ time\rangle$ 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 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.

In curve based trajectory representation model, we represent the trajectory by a sequence of curve segments, rather than of 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

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

where a_0, a_1, a_2 and a_3 are constant coefficients. These 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.

6 ANALYSIS AND 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 the execution of 800 instructions at a sensor node (i.e. a lot of energy is required to transmit a single message). For these reasons the most preferable system is that which requires minimum number of messages 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 messages 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.

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 *LH*s occurs locally, therefore there is no communication cost.
2. **Global Computation Cost**

   a) **Computation Complexity of GH_cs at zero Iteration** For computing the GH_cs at zero Iteration, each node gets the list of *LH*s 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 whereas a node anywhere on the edge has five neighbors, and a node anywhere else has eight neighbors. If we consider $n \times n$ system in which we have n^2 nodes, the total number of neighbors for all nodes in the system will be: $3 \times 4 + 5 \times 4(n-2) + 8 \times (n-2)^2 = 8n^2 - 12n + 4$. Therefore, the number of required messages for this step will be $8n^2 - 12n + 4$.

   b) **Computation Complexity of entire *GH*s**

   - **Computation Complexity of entire *GH*s 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*. So the number of messages needed for this would be the same as before, namely $8n^2 - 12n + 4$. After every iteration a new *GH* component will be added to *GH* or the length of *GH* component increases by either one or more depending on their location in the system and every node replaces its existing *GH* with the new *GH* which was formed during that iteration. Therefore the total number of exchanged messages will be:

     $$Exchanged\ Messages\ = (k+1)(8n^2 - 12n + 4), \tag{4}$$

     where k is the number of iterations $k = n/2 - 1$, this occurs for all nodes in and around the center of the system, where every node has neighboring node on both sides. $k = n - 1$, this occurs for the nodes on the edge and at the corner of the system, where they have neighboring node on only one side to contribute to the growth of their *GH*s during every iteration; the length of their *GH* increases by only one.

- **Computation Complexity 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 requires $8n^2 - 12n + 4$ messages in the worst case, 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, 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 is $8n^2 - 12n + 4$.
  Once all the levels 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 $L \times n_0 \times n_i$. Therefore the total number of messages required to compute the GH for the target node only will be
  $$Exchanged\ Messages = 2(8n^2 + 12n + 4) + L \times n_0 \times n_i. \qquad (5)$$
  From equations 4, and 5 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.

7 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 total number of iterations (this will be applicable to first approach only),
- the location of the target node in the system (this will be applicable to second 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 increases with increasing 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 is at center is less than the number of messages when the target node is at the edge or the corner.

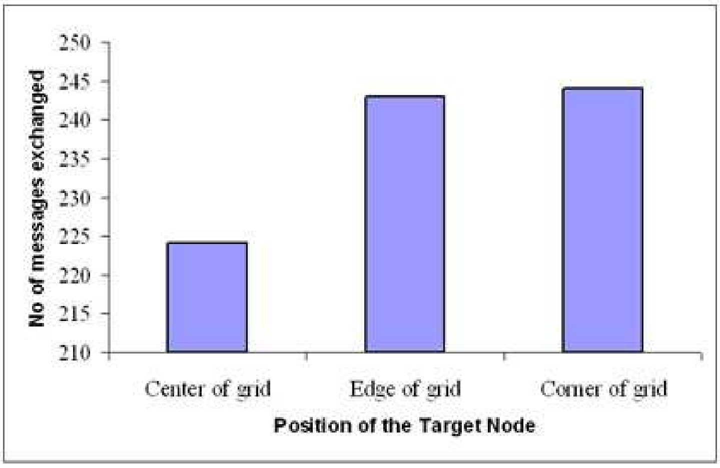

Fig. 2. The number of exchanged messages versus the target node location

- **The total number of iterations:** We found that we require more than one iteration to obtain the entire list of summarized *GH*s. In our implementation we did vary the number of iterations from one to eight, and we found 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.

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.

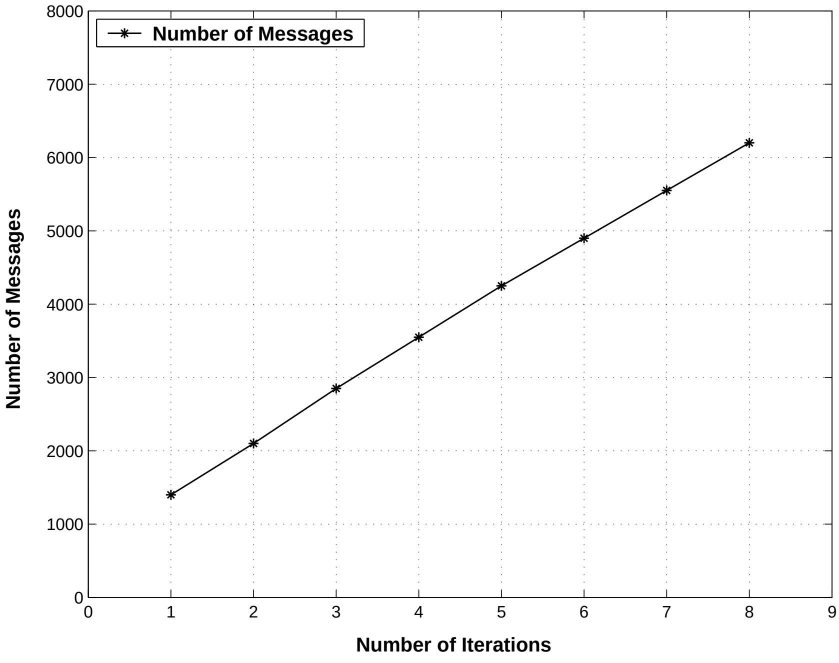

Fig. 3. Exchanged messages versus the number of Iterations – for a node not on edges/corner

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

  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 only 1 611 exchanged messages to form the global entire summarized pattern at target node.

Therefore it is clear that there is significant amount of saving in messages between the two variants, see Figure 4.

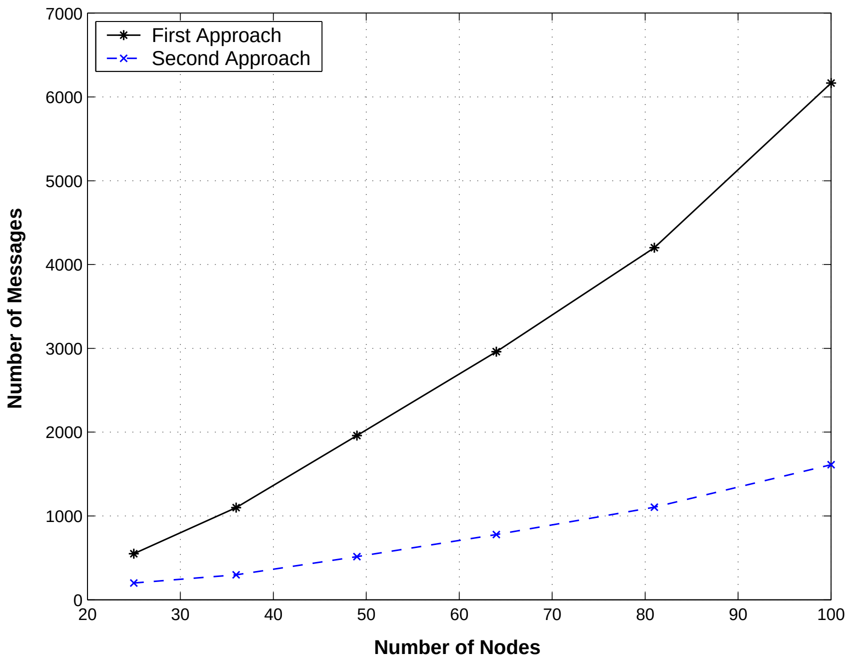

Fig. 4. Exchanged Messages in the two Variants

7.1 Multitargets Tracking

Our mechanism can be readily extended to track multitargets by assuming that each target has a unique ID. Then according to our mechanism the problem will be handled as follows:

1. each sensor forms the local hypothesis LH for each ID. In this case the LH structure will be changed to LH^{id}=(ID, FP, LP, angle)
2. Once the list of LHs of the different targets is collected at the temporary central node, we go through them and form the global hypothesis components GH_c^{id}s for each target by merging its LH^{id}s that lie in the same direction and in asending time stamps.
3. The global hypothesis GH will be formed by merging the global hypothesis components for each target. In this case the GH structure will be changed to (ID, SP, $\langle PointChangeList\rangle$, EP).

8 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 at the beginning, the complexity of our algorithm is 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 lower 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 comparison and contrast of the two variants and presented results obtained for the various test cases for both the variants of our algorithm.

REFERENCES

[1] Agrawal, R.—Shafer, J.: Parallel Mining of Association Rules. In IEEE Trans on Knowledge and Data Engineering, Vol. 8, 1996, No. 6, pp. 962–969.

[2] Aslam, J.—Zack, B.: Florin Constantin, Valentino Crespi, George Cybenko and Daniela Rus: Tracking a Moving Object with a Binary Sensor Network. The First ACM Conference on Embedded Networked Sensor Systems (Sensys 03), Los Angeles, CA, USA, November 2003, pp. 150–161.

[3] Bhaskar, K.—Estrin, D.—Wicker, S.: The Impact of Data Aggregation in Wireless Sensor Networks. TeInternational Workshop in Distributed Event-Based Systems, Austria, 2002, pp. 575–578.

[4] Bongki, M.—Lopez, V.—Fernando, I.—Vijaykumar, I.: Scalable Algorithms for Large-Scale Temporal Aggregation. Technical Report TR 98-11, Tucson, AZ 85721, 1998.

[5] Chen, J.—Yao, K.—Hudson, R.: Source Localization and Beamforming. IEEE Signal Processing Magazine, March 2002.

[6] Chen, W.—Hou, J.—Sha, L.: Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks. 11th IEEE International Conference on Network Protocols (ICNP '03), Atlanta, Georgia, USA, November 2003.

[7] Gupta, R.—Das, S. R.: Tracking Moving Targets in a Smart Sensor Networks. Proc. of VTC Fall 2003 Symposium, October 2003.

[8] Hellerstein, J.—Hong, J.—Madden, S.—Stanek, K.: Beyond Average: Towards Sophisticated Sensing with Queries. 2nd International Workshop on Informa-

tion Processing in Sensor Networks (IPSN '03), Palo Alto, CA, USA, March 2003, pp. 553–569.

[9] Heinzelman, W.—Chandrakasan, A.—Balakrishnan, H.: Energy-Efficient Communication Protocols for Wireless Microsensor Networks. Proc. $33^r d$ Hawaii International Conf. on Systems Science, January 2000.

[10] Intanagonwiwat, C.—Govindan, R.—Estrin, D.: Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networks (MobiCOM 2000), Boston, Massachusetts, August 2000, pp. 56–67.

[11] Khedr, A. M.—Bhatnagar, R.: A Decomposable Algorithm for Minimum Spanning Tree. Distributed Computing-Lecture Notes in Computer Science Springer-Verlag Heidelberg, Volume 2918, 2004, pp. 33–44.

[12] Khedr, A. M.—Bhatnagar, R.: Agents for Integrating Distributed Data for Complex computations. Computing and Informatics, Vol. 26, 2007, pp. 149–170.

[13] Li, D.—Wong, K.—Hu, Y.—Sayeed, A.: Detection, Classification, and Tracking of Targets. In IEEE Signal Processing Magazine, Vol. 19, 2002, No. 2, pp. 17-29.

[14] Liu, J.—Jie, L.—Reich, J.—Cheung, P.—Zhao, F.: Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications. 2nd Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, California, April 2003.

[15] Luo, H.—Ye, F.—Cheng, J.—Lu, S.—Zhang, L.: TTDD: Two-Tier Data Dissemination in Large-scale Sensor Networks. ACM 1-58113-X/02/0009, MobiCOM 2002, Atlanta, Georgia, USA, 2002, pp. 1–13.

[16] Madden, S.—Szewczyk, R.—Franklin, M.—Culler, D.: Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. Workshop on Mobile Computing and Systems Applications, Callicoon, NY, USA, June 2002, pp. 49–58.

[17] Madden, S.—Franklin, M.—Hellerstein, J.—Hong, W.: TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks, Operating Systems Design and Implementation (OSDI). Boston, MA, USA, December 2002.

[18] McErlean, D.—Narayanan, S.: Distributed Detection and Tracking in Sensor Networks. Signals, Systems and Computers, 2002. Conference Record of the Thirty-Sixth Asilomar Conference, Vol. 2, 2002, pp. 3–6.

[19] Mechitov, K.—Sundresh, S.: Cooperative Tracking with Binary-Detection Sensor Networks ACM Sensys '03, 2003.

[20] Pattem, S.—Poduri, S.—Krishnamachari, B.: Energy-Quality Tradeoffs for Target Tracking in Wireless Sensor Networks. 2nd Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, California, April 2003.

[21] Ramanathan, B.: Location-Centric Approach for Collaborative Target Detection, Classification, and Tracking. In Proceedings of IEEE CAS Workshop on Wireless Communication and Networking, Pasadena, California. September, 2002.

[22] Shin, J.—Guibas, L.—Zhao, F.: A Distributed Algorithm for Managing Multi-Target Identities in Wireless Ad-Hoc Sensor Networks. 2nd Workshop on Information Processing in Sensor Networks (IPSN '03), Palo Alto, California, April 2003.

[23] Srikant, R.—Quoc, Vu.—Agrawal, R.: Mining Association Rules with Item Constraints. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, August 1997, pp. 67–73.

[24] Yang, H.—Sikdar, B.: A Protocol for Tracking Mobile Targets using Sensor Networks. Proc. of IEEE workshop on sensor Network Protocols and Applications. Anchogare, AK, May 2003, pp. 71–81.

[25] Zhao, J.—Govindan, R.—Estrin, D.: Computing Aggregates for Monitoring Wireless Sensor Networks. The First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA '03), Anchorage, AK, USA, May 2003, pp. 139–148.

Ahmed M. Khedr received his B. Sc. degree in mathematics in June 1989 and the M. Sc. degree in the area of optimal control in July 1995 both from Zagazig University, Egypt. In March 2003 he received his Ph. D. degree in computer science from University of Cincinnati, Ohio, USA. From March 2003 to January 2004, he was a research assistant professor at ECECS Department University of Cincinnati, USA. From January 2004 till now he is working as faculty at the Mathematics Department, Faculty of Science, Zagazig University, Egypt. He has coauthored 21 works in journals and conferences relating with optimal control, decomposable algorithms, and wireless sensor networks.