DATE: 20th October, 2003

I, RISHI JHAVER, hereby submit this as part of the requirements for the degree of:

M.S

in:

COMPUTER SCIENCE

It is entitled:

Discovery of Linear Trajectories in Geographically Distributed Datasets

Approved by:

Dr. Raj Bhatnagar
Dr. Dharma. P. Agrawal
Dr. Ali Minai
Dr. Karen Davis
DISCOVERY OF LINEAR TRAJECTORIES IN GEOGRAPHICALLY DISTRIBUTED DATASETS

A thesis submitted to the
Division of Research and Advanced Studies
of the University of Cincinnati
in partial fulfillment of the
requirements for the degree of
MASTER OF SCIENCE

in the Department of
Electrical & Computer Engineering and Computer Science
of the College of Engineering
20 October, 2003

by

Rishi Jhaver
B.E., Electrical & Electronics Engineering
Birla Institute of Technology & Science, Pilani, India, 2000

Committee Chair: Dr. Raj K. Bhatnagar
Abstract

We work with temporal data stored in distributed databases that are spread over a region. 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. An algorithm is introduced that mines global temporal patterns from these datasets and results in the discovery of linear trajectories of moving objects under supervision. Each of these datasets has its local temporal dataset along with spatial data and the geographical coordinates of a given object or target. 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. The cost of our algorithm is much smaller than that of the alternative in which the data must be transferred to a single site and then mined.

In addition to this, we vary the requirements of our algorithm slightly and present a variant of it that enhances its performance in terms of the overall complexity of computations. We go on to show that while the efficiency of the algorithm increases in terms of the number of messages exchanged between nodes, the amount of information available to all the nodes in the system decrease. The advantages and drawbacks of this variant of our algorithm is also presented.
Acknowledgements

First and foremost, I would like to thank God who’s always guided me in the right path and showed me the right way in case of any doubts. I would like to express my sincerest gratitude to my advisor, Dr. Raj Bhatnagar for his help and guidance during the course of my thesis work. His continued teaching and constructive ideas and suggestions were very helpful to me in completing my education successfully. Also, I would also like to thank Dr. Dharma Agrawal, Dr. Karen Davis and Dr. Ali Minai for their presence on my thesis committee. I also appreciate their helpful suggestions to improve the thesis. I would like to appreciate the help and support from my colleagues in my lab - Shalini Batra, Shagun Kakkar, Anshumaan Rajshiva, Kaustubh Shinde, Richard Horvitz and Ahmed Khedr. Their continued and cheerful presence made the lab a very good environment to work in. I would also like to thank my other friends from the different departments at the University of Cincinnati who were a source of moral support for me throughout the last two years of my stay here at UC. Last and most importantly, I would like to thank my parents and sister for their ever existing emotional and inspiring support without which I could never have made it this far. I would like to thank them for always being there for me whenever I needed them the most. Thank you everybody.
# Contents

1 Introduction ........................................ 1
   1.1 Data Mining ........................................ 1
   1.2 Sensor Networks .................................... 4
   1.3 Context for Our Problem ............................ 5
   1.4 Organization ....................................... 6

2 Related Research ..................................... 8
   2.1 Data Mining ......................................... 8
   2.2 Sensor Networks .................................... 10

3 Terminology ......................................... 14
   3.1 Introduction ....................................... 14
   3.2 Terms ............................................... 15
      3.2.1 Local Hypothesis .............................. 15
      3.2.2 Global Hypothesis ............................. 16
   3.3 A Second Approach ................................. 18
      3.3.1 Target Node ................................. 19
      3.3.2 Levels ...................................... 19

4 Algorithm Outline .................................. 21
8 Conclusions

8.1 Summary ........................................ 66

8.2 Future Research .............................. 67
List of Figures

3.1 Set of LH for a Sensor Node in a 3 X 3 System . . . . . . . . . . . . . . . . 15
3.2 Set of GH for Sensor Nodes in a 3 X 3 System . . . . . . . . . . . . . . . . 17
3.3 Set of Levels with Target Node in a 10 X 10 System . . . . . . . . . . . . . 19
5.1 List of LHs for Sensor Nodes in a 10 X 10 System . . . . . . . . . . . . . . 37
5.2 List of GHs of length three for Sensor Nodes in a 10 X 10 System . . . . . . 38
5.3 List of GHs of length five for Sensor Nodes in a 10 X 10 System . . . . . . 39
5.4 List of the complete GHs for Sensor Nodes in a 10 X 10 System . . . . . . 40
5.5 List of GHs of length five for Target Node in a 10 X 10 System . . . . . . 41
5.6 List of complete GHs for Target Node in a 10 X 10 System . . . . . . . . . . 42
6.1 Different Type of Placement for Nodes in a 10 X 10 System . . . . . . . . . . 46
7.1 List of GHs for Sensor Nodes in a 5 X 5 System . . . . . . . . . . . . . . . . 56
7.2 List of GHs for Sensor Nodes in a 7 X 7 System . . . . . . . . . . . . . . . . 57
7.3 List of GHs for Sensor Nodes in a 10 X 10 System . . . . . . . . . . . . . . . 58
7.4 List of Levels in a 5 X 5 System . . . . . . . . . . . . . . . . . . . . . . 59
7.5 List of Levels in a 7 X 7 System . . . . . . . . . . . . . . . . . . . . . . 60
7.6 List of Levels in a 10 X 10 System . . . . . . . . . . . . . . . . . . . . . 61
7.7 List of Levels with Target Node Placed on the Edge of the System . . . . . . 62
7.8 List of Levels with Target Node Placed at the Corner of the System . . . . . . 62
Chapter 1

Introduction

1.1 Data Mining

Data Mining is the process of extracting knowledge hidden in large volumes of raw data. The importance of collecting and mining data that reflect business or scientific activities to achieve competitive advantage over others is widely recognized now. Powerful systems for collecting data and managing it in large databases are in place in all large and mid-range companies [1, 2]. However, the main bottleneck of converting this useful information is the design and development of algorithms that extract useful knowledge from this set of data. Human analysts can no longer make sense of enormous volumes of data in order to make informed decisions without the use of specialized tools. Data mining automates the process of finding relationships and patterns in raw data and delivers results that can be either utilized in an automated decision support system or assessed by a human analyst.

Different domains require different kind of data for analyses. These include data such as sales and contacts histories data in the commercial business domain, medical data like patient diagnoses, prescribed drug data and military data such as statistics, regional information. In all
these cases data mining can help us reveal knowledge hidden in data and turn this knowledge into a crucial competitive advantage. Some of the tasks solved by data mining:

a). Predicting:
A task of learning a pattern from examples and using the developed model to predict future values of the target variable.

b). Classification:
A task of finding a function that maps records into one of several discrete classes.

c). Detection of relations:
A task of searching for the most influential independent variables for a selected target variable.

d). Clustering:
A task of identifying groups of records that are similar between themselves but different from the rest of the data. Often, the variables providing the best clustering should be identified as well.

e). Market Basket Analysis:
Processing transactional data in order to find those groups of products that are sold together well. One also searches for directed association rules identifying the best product to be offered with a current selection of purchased products.

f). Deviation Detection:
A task of determining the most significant changes in some key measures of data from previous or expected values.

All of the above scenarios and many more can be easily covered by implementing Data Mining techniques. Some of the methodologies used in Data Mining are: Neural Networks, Evolutionary Programming, Memory Based Reasoning, Decision Trees, Genetic Algorithms, Nonlinear Regression Methods.

Data Mining coupled with Data Warehousing tools is a very potent combination and is being used by a lot of companies for useful and interesting interpretation of historic and archived
In our thesis, we work with sensor networks. We assume that stand alone sensors are spread in a grid like manner in a region to be supervised. They are capable of capturing data and storing them. 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?

Previous research work [34, ?, 9] has proposed a few algorithms for mining of distributed databases. Similarly, we mine distributed datasets for global patterns. Some of the main objectives for designing and developing algorithms for mining of distributed data are:

1). They should be able to decompose themselves onto a single data site efficiently;
2). There should be minimum communication between the different data sites;
3). The communication between the data sites should be secure.

There has been little work that deal with the design and developing of algorithms for mining data in sensor networks [6, 12, 14] while taking into account the above objectives. Our work develops an algorithm that address the first two of the three points. We do not take into the account the problem of security and that is a possible future research area. We work with distributed databases spread over a geographical region and are capable of storing of data as in a database/dataset. This could include commercial data, data used in the prediction of weather or data collected by sensor networks in a battlefield. We have taken into account the trajectory
of a moving object through a region of sensor nodes. This trajectory could be of any object like a truck, person or any vehicle. We have designed and developed an algorithm that considers data from many distributed sensor nodes and computes global patterns for all of them. The main objective of our algorithm is to compute the summarized global patterns by doing in-network aggregation in the sensor network.

1.2 Sensor Networks

Data Mining techniques are used in combination with Sensor Networks for a host of applications. For the purpose of our thesis, we refer to sensor networks as a set of geographically distributed standalone sensors that are spread over a region and are capable of sensing predefined parameters such as temperature, motion and geographic coordinates of objects. 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. We assume that the sensors we work with 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 thesis work to stationary sensors. It is an area of future research to design and develop algorithms for mobile sensors. We now list some of the applications of sensor networks.

Applications of Sensor Networks:

A number of emerging problems require handling of this type of data requiring pattern recognition.

i). An example is the discovery of interesting socio-economic temporal patterns across the economic databases maintained by each county in a state.

ii). A second example could be a network designed to monitor a power utility system, in which
sensors are deployed at meter interval along every pipe. Each sensor can sample the flow rate every few seconds.

iii). Another example is the use of distributed sensors in a forest or a wild life sanctuary which is to be studied for the movement of wild life. Motion sensors can be deployed in the area which would record any motion along with the timestamp. At the end of day, this data can be analyzed for trends in movements of wild life or any other required information.

1.3 Context for Our Problem

Sensor networks can be deployed in a number of real time applications to obtain useful patterns in data which could be used for a variety of purposes. But there are some problems and challenges in using distributed sensor networks:

1). Sensor networks usually require a high amount of exchange of data between sensors. In case of a lot of data, this creates problems because communication bandwidth is a key constraint in any sensor network. Communication bandwidth available to sensors in a network is very limited and the algorithm implementing data mining has to take into account this and minimize the number of exchange of messages.

2). 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. Options include moving-window averages or collaborative processing among clusters of sensor nodes to detect events or features that have spatio-temporal content.

3). If the amount of data is very high and varies with time, it might require a second (or more than two) scanning of the databases in the sensors to get all the data in memory. Usually, single-pass algorithms are preferred. This problem is addressed by our algorithm.

In our thesis, we present an algorithm for mining distributed databases for discovering temporal patterns. The main objective of this algorithm is to minimize the number of exchanged messages among the participating databases (or sensor nodes) and maximize the number of computations at each local data site (or sensor node). From the summaries generated at the local site, a further round of exchange of messages is done and global patterns are discovered.

1.4 Organization

The organization of our thesis is as follows:
Chapter 2 gives an idea of related research done in the area of Data Mining in Sensor Net-
works. In Chapter 3, we introduce some basic terms used in our thesis and also define the problem statement. Next, we introduce the algorithm and describe the steps involved in it in Chapter 4. Chapter 5 illustrates our algorithm with an example. In Chapter 6, we discuss and analyze the complexity our algorithm. In Chapter 7, we present the test results run for different test cases for the different variants of our algorithm. In the final Chapter, we discuss future research area and end with the conclusions.
Chapter 2

Related Research

In this chapter, we look at some of the related research work done in the area of our thesis. We compare these with our thesis and state the difference between the two. Papers from data mining, sensor networks, distributed and parallel mining have been looked at and analyzed.

2.1 Data Mining

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 [1, 2, 3, 27, 19, 20]. 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 DBMiner system) whereas the latter discovers strong regularities or association rules from databases without concept generalization.

One of the main KDD tasks is the discovery of association rules. The work done by [26]
introduces the problem of mining generalized association rules from a large database of trans-
actions. The main problem of mining association rules [1] is that of finding all the rules that
satisfy a user-specified minimum support and minimum confidence. In general, algorithms
for data mining and knowledge discovery have received lot of attention [26, 32, 27, 1, 2] in
the recent past. Algorithms for fast association mining are discussed by [32]. The most im-
portant measurable criterion in these algorithms is the number of database scans required to
determine all the association rules. In this paper, the entire database is scanned only once
which considerably reduces the overhead due to repeated database scanning. Related database
sets are divided into clusters which facilitate faster processing of the association rules. But
in practice, most of the users are interested in a subset of association rules only. This can be
achieved by applying relevant constraints as a post processing step to the association rules as
is explained by [27]. They implement a few algorithms which integrate constraints into the
normal data mining algorithm. However, all these algorithms are designed for databases that
typically reside as single datasets or tables in data warehouses.

It is fair to say that many sequential algorithms have been proposed for mining of associ-
ation rules. The most popular ones are Apriori, DHP and Partition algorithms [2, 19]. In the
Apriori algorithm, a candidate set generation function is adopted which supports an efficient
method of generating candidate sets which lead to formation of association rules. DHP uses a
hashing technique to prune away irrelevant candidate sets to improve its efficiency. Partition
divides the database into small partitions so that they could be easily processed in memory
independently to find out their large and final itemsets.

However, little work has been done comparatively in mining association rules in distributed
databases. If the algorithms used for sequential algorithms were to be directly used for dis-
tributed databases, they would involve the overhead of very high communication costs. Hence
separate algorithms need to be designed which can be implemented in distributed databases.
An introductory idea about distributed algorithms is given by [34]. An optimum and efficient algorithm is distributed mining of association rules (DMA) [9] which could mine horizontally distributed databases by generating a small number of candidate sets. The complexity in this paper was found to be linear with the number of candidate sets which turns out to be very efficient.

2.2 Sensor Networks

A very important and often discussed application of distributed algorithms for sensor networks is environmental monitoring. 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 [24]. This is subsequently used for effective monitoring of the environment. Also, an error estimate is drawn and the complexity of the algorithm is discussed. There is a lot of work going on in this area at UC Berkeley (Sensorwebs and Smart Dust projects). Some previous work on environmental monitoring using random sensor networks was done by [12]. This paper is different from our work in that the gradient between nodes is calculated as a difference between the measure of the given scalar variable between the two nodes. But in our work, we calculate the gradient of our network as the distance between the sensor nodes in the system and the predefined target node.

There has been a lot of increased research activity in Sensor Networks over the last few years, with advances in sensor node and radio hardware [25, 21]. The balancing between computation and communication has been explained in this work and a lot of work in In-network processing also has been done. This is one of the motivating papers for our research in in-network processing. The topographically addressed sensor nodes are similar to the way we have dealt with the placement of nodes in our work.
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 [6]. [22] 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. This paper differs from traditional network systems in terms of its structure and architecture.

Research [4, 28] has 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 [22]. In this aspect, the central idea of our thesis is similar with [22]. 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. Algorithms have been designed [14] 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 [14] 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. In our paper, we assume that data flows between nodes from a higher level to a lower level only.

There have also been recent research work on adaptive web caching [17] and peer-to-peer file sharing systems such as Freenet [10] that explain application specific hop-by-hop processing. A particular application specific protocol is used in each of these papers which is different from our thesis. Also, our work runs directly over hop-by-hop network rather than a virtual network.

Ad-hoc routing (such as DSR and AODV) can be used in sensor networks [7]. The recent advances in technology have made it feasible to build ad-hoc sensor networks using inexpensive nodes which would consist of low power, a modest amount of memory, a wireless transmitter [15]. There are many real world applications for an ad-hoc network like habitat monitoring, target tracking, meteorological applications, etc. To locate sensors in an ad hoc network, we would need some kind of directory look up service. An algorithm is discussed by [13] 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. Another contrasting point in this paper is that the messages flow from the center of the network to the periphery. In our work, gradients are set up based on the target node and subsequent messages flow from the periphery (higher gradient) to the center of the network (lower gradient).

Jini is another such example [29]. 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 thesis is fundamentally different from these works 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. Another method to locate sensor nodes is by using XML-based objects through a network of collaborative servers which is done by the Ninja Service Directory Service [11]. The overhead here is the excessive bandwidth required.

A new mechanism for routing based on clusters called LEACH is proposed by [13]. Intermediate range communication can be done via cluster heads and the subsequent in-network data compression can reduce energy consumption in the network. This paper stresses more on the conservation of energy in the network. The specific details of the flow of in-network data aggregation has not been specified. But our work is in line with the kind of work done here. In our thesis, the emphasis is more on data analysis in the network so that the desired global patterns can be obtained. In-network aggregation helps us reduce the complexity of our algorithm.

In our paper, we discuss an algorithm which discovers global temporal patterns among distributed databases. For our problem formulation, we need algorithms that can decompose themselves into partial computations to match the network locations of data components and execute these at the participating nodes, and compose the received partial results to construct global results.
Chapter 3

Terminology

3.1 Introduction

The main aim of our algorithm is to discover global linear patterns among a set of distributed datasets. We refer to sensors or nodes as the individual sensor nodes which are placed in the system and are capable of collecting and storing data. We work with intelligent sensors which are capable of measuring a scalar quantity or variable like the intensity of light, movement, sound, etc. The development of our algorithm is in the context of temporal data being recorded by these sensors in the system. We assume that these sensors are spread evenly in a square network which we also call as a grid. Sensors are capable of local as well as global processing and exchanging of data. The main aim of our algorithm is to develop summarized global patterns in a system of nodes. Before proceeding further, we need to understand what a target (moving) object is. We take the example of a battle field where many sensor nodes have been spread all over the area. We assume that these sensors have the capability to capture the information of any moving object in its vicinity or sensing range. This object could be an enemy truck or a tank or even a person. Over a period of time, the sensors capture detailed information about such target objects (which can be specified by the user). The information
contain the approximate X, Y coordinates, the timestamp and any other local variable like temperature, motion, speed, etc. This information or data is used in the generation of summarized global patterns. We refer to the target object as a moving object throughout our thesis. We now outline the basic terminology used in our work.

### 3.2 Terms

#### 3.2.1 Local Hypothesis

![Set of LH for a Sensor Node](image)

**Figure 3.1: Set of LH for a Sensor Node in a 3 X 3 System**

Forming of Local Hypotheses is the first step that is implemented in each of the sensor nodes. We refer a point to a set of X, Y and T coordinates recorded by the sensor. A sensor records points as and when a moving object passes through its sensing range. These set of points are the data present in each of the sensors. Upon collection of the required data, it is analyzed at the local site for possible patterns. We refer to these patterns as Local Hypotheses. A Local Hypothesis refers to patterns in the local node and is formed by 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.

If we consider four points, \( p_1, p_2, p_3 \) and \( p_4 \), then we define the angle between \( p_1 \) and \( p_2 \) as:
\[
\text{Angle}(p_1, p_2) = \tan^{-1} \left( \frac{\text{Ycoord}(p_2) - \text{Ycoord}(p_1)}{\text{Xcoord}(p_2) - \text{Xcoord}(p_1)} \right).
\]

In a similar way, we can find the angles between any pair of points. 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 other points. The angle can vary from 0° to 360°. In Fig 3.1 we consider a system of nine sensor nodes placed in 3 X 3 fashion. In node five, \( t_1, t_2, t_3, t_4 \) all lie in the same direction (they’re in the same line). Also they are in ascending timestamp fashion with the timestamp chronologically increasing from \( t_1 \) to \( t_4 \). Hence these four points constitute one Local Hypothesis for a sensor node. This is shown by outlining the points with a rectangular box in Fig 3.1. In this way, we form LHs for every node.

### 3.2.2 Global Hypothesis

We refer to Global Hypotheses as GHs. Forming of GH is the second step of our algorithm. Forming of a Global Hypothesis is similar to that of Local Hypothesis. A GH is formed when LHs from three or more sensor nodes are merged to form a set of points that starts with the first point in the first node and ends with the last point of the final node and passing through the second node. It can be considered as a global Local Hypothesis. The criteria for merging the points from different nodes are the same as in the previous step but they are applied to neighboring nodes instead.

We define the length of a GH to be the number of nodes considered for forming the GH.
If points from three nodes have been merged, the length of the GH is three. In a more generic way, a GH of length n would mean that points from n nodes have been checked for the above defined criteria and have been merged to form the GH. In this way, a GH is formed.

The direction of a GH is obtained from the angle component in it. A GH constitutes three LHs. The angle component in the LH corresponds to the angle or the direction of the GH. The direction of a global hypothesis could be in any angle (from 0° - 360°) as is with the LHs.

In Fig 3.2, we consider a system of nine nodes in a 3 X 3 fashion; the LH from node 4 is $t_1, t_{21}, t_3$ and $t_4$; LH from node 5 is $t_6, t_7, t_8$ and $t_9$ and LH from node 6 is $t_{11}, t_{12}, t_{13}$ and $t_{14}$. These points lie in the same line in the same direction (the angle formed by all of these points is 0°) and are in ascending time stamp manner from node 4 to node 6. They satisfy the criteria and hence we can combine these points to form a GH of length three since points from three nodes have been considered; nodes four, five and six. The resulting GH or summarized pattern would be $t_1, t_2, \ldots, t_{14}$ as is shown with a rectangular box in the Fig 3.2.
A very important point here is that we consider Global Hypotheses for nodes in a linear direction only. If the moving object changes direction while in the system and then moves in the same direction, this would be seen as two Global Hypotheses and not one. As is shown in the Fig 3.2, the GH2 and GH3 actually correspond to the same object moving in different directions. But due to the limitation of our algorithm, this is seen as two GHs for that moving object. To combine these two into the same GH would be an area of future research.

3.3 A Second Approach

Till now, we have defined and described the idea of computing the global hypotheses for all the sensor nodes in the system. We can view a GH as a pattern spread over a collection of nodes. Also, we can say that a GH for a node is a pattern of points which goes through itself. In our algorithm, a sensor node can have more than one GH passing through itself and can compute all of its own GHs. The constraint here is that a node can only compute its own GHs. In other words, it can find out only those GHs which pass through itself. If a GH does not involve that node, it does not have any knowledge of it. For e.g, a node in GH1 in Fig 3.2 does not have any knowledge of GH2 and vice versa. The GHs are calculated for all the nodes but this knowledge is limited to that particular node only.

We measure the complexity of our algorithm in terms of the number of messages exchanged between the nodes to compute the final GHs for the nodes. The communication costs involved in this process is very high, because of which the complexity of our algorithm in this case (of determining the GHs for all the nodes in the system) becomes very high. Hence we modify our requirements such that the global hypotheses for only a single sensor node is required and not for all the nodes in the system. This results in a much lower complexity for our algorithm. We introduce the concept of a target node.
3.3.1 Target Node

A target node is a predefined designated sensor node whose summarized global hypothesis is to be computed. After deciding a target node, our algorithm is implemented. The complexity of our algorithm reduces drastically in this case; reduces by almost 67% when compared with the complexity of the previous case in which we compute the GHs for all the nodes in the system. However, there is a trade off involved here; we do away with the knowledge of the GHs for the other n-1 nodes in the system but gain on the performance side. Our algorithm is adaptable to both the situations. The target node could be chosen as any one from the n nodes in the system. In the Fig 3.3, the target node is shown as the central most node of our 10 X 10 network.

3.3.2 Levels

After deciding a target node, we divide the entire system into levels based on the proximity of the nodes from the target node. We assume that the target node is at level zero. Its immediate
neighbors are set at level one and their immediate neighbors are assigned a level two. In this case the level number keeps increasing by one till all the nodes in the system are covered. We see from Fig 3.3 that the immediate neighbors of the target node are at level 1 and the other nodes are at their respective levels. All the five levels can be seen in the Fig 3.3. We restrict the communication between nodes from higher level nodes to lower level nodes only. This means that nodes at a higher level only, can send messages to those at a lower level; exchange of messages from nodes in the lower level to those at a higher level is not allowed. In this way, we save a lot on communication costs. The number of exchange of messages or LHs is drastically reduced which in turn reduces the complexity of our algorithm.

Again from Fig 3.3, we see that only the nodes from level five can send messages to nodes in level four as is depicted by the inward arrow from level five in to level four; the other way around is not allowed. In this there is a flow of messages from the periphery of the system to the center of the system which is represented by the predefined target node. At every level, the messages received from the nodes at the higher level is merged with the information in the current node and is then sent to the nodes at the lower level. The target node receives the summarized messages from all its neighbors and simply merges them based on the criteria to form the resultant summarized global pattern. In this way, we increase the efficiency and performance of our algorithm by assigning levels to all the nodes in the system and by allowing only one way communication in the system.
Chapter 4

Algorithm Outline

4.1 Introduction

In this chapter, we present a step by step outline of our algorithm with every step described in
detail. The crux of our algorithm is based on the fact that the the cost of processing data at a
local site is very less compared to the cost of exchanging messages/data between nodes. We
implement this idea very effectively in our algorithm by maximizing the processing of data
at the local sensor node and minimizing the exchange of information or messages between
nodes. We need to have some exchange of information between nodes in order to arrive at the
summarized global patterns; albeit we try to minimize that number.

4.2 Organization

We first explain the steps of computing the Local Hypotheses (LH) and the Global Hypotheses
(GH) for all the sensor nodes in the system. In this, we assume that the user is interested in
obtaining information about the Local Hypotheses and the Global Hypotheses for all the nodes
in the system.
Then we describe the second phase of our algorithm where we reduce the complexity of our algorithm and increase its efficiency and performance by choosing a target node and computing the summarized patterns only for that node. At the start of every section, we present the pseudo code for that step which explains the working of our algorithm.

Our algorithm is developed and tested in Java. The four main methods in our algorithm are:

- **formLH()** computes the list of Local Hypotheses for all the nodes, locally;
- **generateGH()** forms the list of Global Hypotheses of length three for all the nodes in the system;
- **computeGH_AllNodes()** computes the list of Global Hypotheses for all the nodes in the system;
- **computeGH_TargetNode()** computes the list of Global Hypotheses for the target node only.

A step by step description of this code is given after the pseudo code in each section. Also, we explain our algorithm with the help of an example in the next chapter. We start with the first method where we compute the list of LHs for all the nodes in the system.

### 4.2.1 Computations of Local Hypotheses

We refer to the term point often in this chapter. It refers to a combined set of information about the moving object which includes the X, Y coordinates and the timestamp. We assume that the system clock for all the nodes in the system are synchronized and hence show a universal time with no mismatches. We also assume that the approximate X and Y coordinates are already known to the sensor nodes through some kind of GPS mechanism. Also, we assume
that for each sensor node, there is a region ‘R’ (also called the range of the sensor node) in which it can sense any kind of motion or other scalar variable like sound, temperature, etc and can record the details of that object. The range of all the nodes are assumed to be mutually exclusive of each other. If we assume three points $p_1$, $p_2$ and $p_3$, we say that the three points lie in ascending manner of timestamp if the timestamp of $p_1$ is before $p_2$ and that of $p_2$ is before $p_3$.

**Pseudo Code:**

1). procedure formLH()

**Output:** set of Local Hypotheses for all the individual sensor nodes in the system

1. Take a set of three points $p_1$, $p_2$ and $p_3$
2. For (every set of $p_1$, $p_2$ and $p_3$ in the database of the local sensor node)
3. Find the angle: $\text{Angle}_1 = \text{angle}(p_1,p_2)$ and $\text{Angle}_2 = \text{angle}(p_2,p_3)$
4. If ($\text{Angle}_1 == \text{Angle}_2$) AND (the timestamps of $p_1$, $p_2$ and $p_3$ are in ascending manner)
5. Add points $p_1$, $p_2$ and $p_3$ to an LH, $L_i = (p_1,p_2,p_3)$ in a data structure (3D array)
6. Take the next point $p_4$ in the database
7. If ($\text{angle}(p_1,p_4) = \text{angle}(p_1,p_2)$ AND If (the timestamps of $p_1$, $p_2$, $p_3$ and $p_4$ are in ascending manner)
8. ADD fourth point $p_4$ to the Local Hypothesis $L_i$
9. Else GO TO the next point $p_5$
10. Else
11. Go to next set of 3 points - $p_1$, $p_2$ and $p_4$

When an object moves in the sensing range of a node, it records the details of every point of the object. We assume that the battery life is sufficient to record the details the object. Also, we assume that if the battery goes dead, there is a replacement battery which takes over. In this way, every sensor node in the system captures the details like X, Y coordinates and the timestamp of every point of the moving object. We assume that the frequency of recording the details of the object is given by the user and can be adjusted accordingly. At the end of a time period, the database of each of the sensor nodes is scanned for these values and is stored in
memory for further computing of the Local Hypotheses and then the Global Hypotheses. This
time period can be fixed by the user.

The details of all the points (X,Y coordinates and Timestamp) are then sorted in ascending
order of their timestamps. Once this is done, all the points in the database are checked for the
following criteria for forming of Local Hypotheses.
1). Check whether the points lie in the same direction and on the same line
2). Check whether the timestamp of the points are in ascending timestamp fashion

The LHs (Local Hypotheses) for a node are formed based on the above criteria. In the descrip-
tion given below, we refer to the pseudo code by referring to its line numbers. A minimum of
three nodes is necessary to form an LH (line 3 and 4). The data (X,Y coordinates and Times-
tamp) present in each of the nodes is checked for these criteria (line 7); upon satisfaction, the
LHs are computed for the node by merging all the points and storing them as one set of data
in memory (line 8). We merge the data by combining the details of all the points into a single
set of details and refer to that set as a Local Hypothesis. For e.g., if the three sets of points
are \((x_1, y_1 \text{ and } t_1), (x_2, y_2 \text{ and } t_2)\) and \((x_3, y_3 \text{ and } t_3)\), then we first check the angle formed
between these points by using their X and Y coordinates. If the angles are the same, we check
if their timestamps are in ascending/descending timestamp manner or not. If they are, then
we merge these three points into one resultant point - \((x_1, x_2, x_3), (y_1, y_2, y_3)\) and \((t_1, t_2, t_3)\).
This corresponds to one Local Hypothesis for the sensor node. The length of an LH can be
more than three also. Hence we check for other points in the node’s database taken one at a
time (line 9). If this points satisfies the criteria (line 10), it is added to the same LH (line 11)
otherwise the next available point in the database is checked. In this way, all the points in the
database are exhausted. In this manner, we compute an LH for a node.

In a similar way, the LHs for all the nodes in the system (line 14) are computed and stored at
the local site. Another point to be noted here is that the minimum number of points to form a LH is three. This is also checked prior to the forming of LHs.

In this way, by taking sets of three points at a time and checking them for the criteria satisfaction and then adding a point subsequently, the set of Local Hypotheses for every sensor node is formed. The length of a LH in a node can be as high as the total number of points in the node’s database along the same direction and can be as low as three.

4.2.2 Computations of Global Hypotheses - of Length Three

Every node in the system is associated with an index number which indicated 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. We refer to a neighbor of a node $N_i$ as $N_{ngh}$. 
2). procedure generateGH()

Output: set of Global Hypotheses of length three for all the nodes in the system

1. For (all the nodes $N_i$ in the system taken one by one)
2. For (all the neighboring nodes of $N_i$: $Ngh_i$)
3. Take the set of Local Hypothesis $LHi$ for $Ngh_i$
4. Add $LH_i$ to a data structure - $DS_i$ of $N_i$
5. Divide the data structure $DS_i$ which stores the entire list of LHs into two 3D arrays $DS_1$ and $DS_2$ for each Node Ni.
6. Every node has $DS_1$ and $DS_2$
7. For (Take an LHi from $DS_1$ and an LHj from $DS_2$)
8. Find the angle component in the two LHs; angle($LH_i$) and angle($LH_j$)
9. If (the two angles are the same) and IF (timestamp of $LH_i$ and $LH_j$ are in ascending manner)
10. Add $LH_i$, $LH_j$ into a data structure $GH_i$
11. Go to the next LH in $DS_2$
12. Else
13. Go to the next pair of LHs ($LH_i$ and $LH_j$) in $DS_1$ and $DS_2$

Once the knowledge of the LHs for each node as well as its neighbor’s is obtained by every node, it is ready to perform computations on that list of LHs to form a Global Hypothesis. In the first step, all the neighbors of a node send their list of LHs to that particular node (lines 3-6). This process is repeated for all the nodes in the system. At the end of this step, all the nodes in the system have information of their LHs as well as those of their neighbors. This is used in the forming of the GHs. As in the previous case, the minimum number of nodes to be taken into account to form a GH for a node is three.

By now every node has the following stored in a data structure: list of its own LHs as well as list of LHs of its neighboring nodes. We denote that data structure as $D_s$. In the next step, each node divides $D_s$ into two parts; $D_{s1}$ and $D_{s2}$ based on the index number of the current node (lines 7,8). Hence the LHs of all those nodes which have an index number less than or equal
to the current node number will be in $D_{s1}$ and the remaining LHs are stored in $D_{s2}$. These two
data structures are then used in the following steps to compute the GHs.

As in the previous section, we define three $p_1, p_2$ and $p_3$ to be in ascending order of timestamps
if the timestamp of $p_1$ is before that of $p_2$ and the timestamp of $p_2$ is before that of $p_3$. Also,
we define the angle of a Local Hypothesis in the following way. An LH comprises of three or
more points which satisfy certain criteria, as is mentioned in the previous section. The angle
at which these points lie with respect to each other is defined as the angle of the LH. This is
measured by using the geometric definition of an angle between 2 points. If we consider two
points, $p_1$ and $p_2$, we define the angle between $p_1$ and $p_2$ as

$$\text{Angle}(p_1, p_2) = \tan^{-1} \left[ \frac{\text{Ycoord}(p_2) - \text{Ycoord}(p_1)}{\text{Xcoord}(p_2) - \text{Xcoord}(p_1)} \right].$$

Every node goes through its list of $D_{s1}$ and $D_{s2}$ and picks up an LH from each of them
(line 9). Let us denote those two LHs as $LH_1$ and $LH_2$. It then checks these two LHs for
satisfaction of the criteria

- Check whether the points in the two LHs of the nodes lie in the same direction and on
  the same line (lines 10,11);
- Check whether the timestamp of the points in the two LHs of the nodes are in ascending
  timestamp fashion (line 12). If they satisfy, then they are merged into one single Global
  Hypothesis (line 13).

In other words:

If $\text{Angle}[LH_1] = \text{Angle}[LH_2]$ and
If $\text{Timestamp}[LH_2], \text{Timestamp}[LH_2]$ are in ascending manner,

they are merged as $(LH_1, LH_2)$ - a single GH.
Once the GH is formed, we can say that the length of this GH is three. This length indicates the number of nodes taken into account to form the Global Hypothesis. After the formation of a GH for a node, it is possible that there are more GHs available for the same node, but in a different direction. Hence, we exhaust all combinations of LHs from $D_{a1}$ and $D_{a2}$ to arrive at all possible GHs for that node (lines 15,16).

As in the previous step, this step is repeated for all the nodes in the system. At the end, every node has a list of GHs of length three (conditional to the fact that the GHs for the nodes do exist). Sometimes it is possible that the points in the LHs of the nodes do not satisfy the criteria. In this case, there is no GH formed for that particular node. This marks the end of iteration zero.

We say that at the end of the zeroth iteration, all the nodes in the system have knowledge of their GHs which are of length three. These iterations can be repeated if required and at the end of every iteration the length of the GHs increase by one or two depending on the location of the node in the system. If the node is located at the edge or the border of the system, its GH’s length increases by one else the length increases by two. We discuss this in more detail in the following section.

In our algorithm, after the zeroth iteration, we have two options: We can either compute the entire global summarized patterns (GHs) for all the nodes in the system or compute the entire global summarized patterns (GHs) for a pre-specified target node only. Depending on the choice we make, we go ahead with one of the next two steps.
4.2.3 Entire Global Computations For All Nodes

Pseudo Code:

3). procedure computeGH_AllNodes()

Output: set of entire global hypotheses formed by the all the nodes in the system

1. For (number of iterations required)

2. For (all the nodes in the system taken one at a time - \( N_i \))

3. For ( all the neighboring nodes of \( N_i \))

4. Take a node \( N_i \) and its neighboring node \( N_{ngh} \)

5. For( \( \text{GH}(N_i) \) and \( \text{GH}(N_{ngh}) \))

6. If (\( \text{Angle}(\text{GH}(N_i)) = \text{Angle}(\text{GH}(N_{ngh})) \) and \( \text{Timestamps}(\text{GH}(N_i)), \text{Timestamps}(\text{GH}(N_{ngh})) \) are in ascending timestamp order)

7. Merge the two GHs into one GH; \( \text{GH}_{\text{resultant}} = [\text{GH}(N_i), \text{GH}(N_{ngh})] \)

8. Add the consolidated GH - \( \text{GH}_{\text{resultant}} \) to a common data structure; this keeps getting updated during every iteration}

9. Else

10. Gto the next GH(\(N_{ngh}\))

11. Goto the GHs obtained from \( N_i \) and the next neighboring node

In this step, we compute the consolidated Global Hypotheses for all the nodes in the system. The GHs computed for the nodes from the previous step are of length three. It is possible that the actual summarized global pattern could cover more nodes. In order for us to discover that actual pattern, we need to repeat the procedure of computing the GHs as done in the previous step.

Let us consider a GH1 of length three from a node to be \((l_1, l_2, l_3)\) and a GH2 from a neighboring node to be \((l_2, l_3, l_4)\). The angle of this GH would be the angle component of the individual LH, which the GH consists of. Hence the angle of GH1 is the angle corresponding to one of \(l_1, l_2\) or \(l_3\). In a similar way we find the angle for the other GHs in the system. The timestamps
of a GH correspond to the timestamp of the points in the various LHs of that GH. With these definitions, we proceed to compute the entire GHs for all the nodes in the system.

The list of GHs for every node is checked with the list of GHs obtained from the neighboring nodes for the criteria mentioned in the above sections. In lines 1-3, we decide the number of iterations to be performed. This decides the length of the eventual GH and is discussed later on in this section. Then, we consider all the nodes in the system one by one in Line 2. We get the GHs of all the neighboring nodes for a node (lines 3,4) and then compare them for possible satisfaction of the criteria. If they are at the same angle and if the timestamps are in ascending manner (line 6), we merge these two GHs from a node and one of its neighboring node respectively to form a GH of an increased length (lines 7,8). For e.g. if GH from a node was \((l_1,l_2,l_3)\) and the GH from the neighboring node was \((l_3,l_4,l_5)\), then the resultant GH by merging these two would be \((l_1,l_2,l_3,l_4,l_5)\). This represents a GH obtained by merging the GHs of two nodes. The length of this GH is more than the length of the individual GH, previously. Once this is done, we check the GHs for the same node with the GHs from its next neighboring node for possible GHs of an increased length. In this way, we check all the neighbors for this node.

In this way, we take the list of GHs from every pair of nodes in the system and combine them based on the criteria to form a resultant list of GHs. This is done for all the nodes in the system and this whole procedure constitutes one iteration. The number of iterations is fixed by the user and after every iteration, the length of the GH gets increased till it reaches its final length. For instance, if after iteration one, the length of the GH is 5, its neighbors’ length is also five, then after the second iteration, the length of the GH for the node becomes 6 or 7 depending on its location. This is explained in more detail in Chapter 7. The length for any sensor node is dependent on the number of iterations carried out. We assume that the number of iterations carried out is sufficient to let every node compute its list of entire and complete
In this way, we compute the complete GHs for all the nodes in the system. In the worst case scenario, the number of iterations required by the system is equal to the square root of the number of nodes in the system. The number of iterations can be related to the length of the Global Hypothesis of the node. If we assume the length of the GH to be \( L \) and the number of iterations to be \( I \), then

\[
L = (I*1) + 1 \text{ for all the nodes on the edge/border of the network}
\]
\[
L = (I*2) + 1 \text{ for all the other nodes in the system}
\]

Thus, we can find the length of the Global Hypothesis at any given iteration. Thus we compute the summarized global patterns for all the sensor nodes in the system. The computations for the global patterns for only the target node is explained in the following step. Also, the complexity of our algorithm is discussed in Chapter 6.
4.2.4 Global Computations For Variant Of Algorithm

<table>
<thead>
<tr>
<th>4). procedure computeGH_TargetNode()</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Output:</strong> set of nodes with respective levels assigned based on its distance from the target node</td>
</tr>
<tr>
<td>1. Assign 0 as the level for the target node</td>
</tr>
<tr>
<td>2. <strong>For</strong> (all the nodes in the system except the target node)</td>
</tr>
<tr>
<td>3. Compute proximity of node from target node based on coords of node</td>
</tr>
<tr>
<td>4. Assign level based on distance; level number increases by one as dist. increases</td>
</tr>
<tr>
<td>5. <strong>For</strong> (levels in the system from the highest to zero)</td>
</tr>
<tr>
<td>6. <strong>For</strong> (all the nodes in the level)</td>
</tr>
<tr>
<td>7. Receive list of consolidated GHs from neighboring node at higher level</td>
</tr>
<tr>
<td>8. Check if the criteria are satisfied for the two lists of GHs</td>
</tr>
<tr>
<td>9. If Criteria satisfied Then merge the two GHs and store for current node</td>
</tr>
<tr>
<td>10. If Criteria not satisfied, Then do not merge the two lists of GHs; keep the original list of GHs as it is</td>
</tr>
<tr>
<td>11. Check the level of all the neighbors</td>
</tr>
<tr>
<td>12. If level is lower, send list of consolidated GHs to neighboring node</td>
</tr>
<tr>
<td>13. Go to the next neighboring node</td>
</tr>
<tr>
<td>14. Else go to next neighboring node</td>
</tr>
<tr>
<td>15. If node is Target Node, then simply merge lists of GHs after checking for the criteria</td>
</tr>
<tr>
<td>16. Do not send consolidated list of GHs to any other sensor node. This resultant list of GHs is the desired output</td>
</tr>
</tbody>
</table>

We introduce this step as a second variant of our algorithm. In this, the complexity of our algorithm is reduced drastically as is shown in Chapter 6. We assume that the summarized GHs are 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. Also, we assume that all the nodes in the system have knowledge of the target node before they implement our algorithm. By incorporating a very few lines of code, each node in the system can intimated of the target node. Once this is done, the next step is to compute the levels for the
The target node is assigned zero level (line 1). This is the lower most level in the system. Next the neighboring nodes of the target node are assigned a level one. The level of the nodes at every step is incremented by one. The next neighboring nodes is assigned a level of two and in this way all the nodes in the system are assigned their respective levels (lines 2,3,4).

Once the assigning of levels to all the nodes is done, the next step is the exchange of messages between the nodes to form the summarized global pattern. However the following constraints exist while sending messages to the neighboring nodes.

- Each node can send messages (or its list of GHs) to its neighboring nodes only. Nodes at a higher level can send messages to nodes in a lower level
- Nodes at a lower level can receive messages from nodes at a higher level but cannot send messages to those nodes

Thus, in this way, nodes at the highest level start sending their list of GHs to their neighboring nodes which lie at a lower level. As in the previous section, the basic definition of the angle of GH and timestamp of GH remains the same. Let us consider a $GH_h$ of length three from a node at a higher level to be $(l_1, l_2, l_3)$ and a $GH_l$ from a neighboring node at a lower level to be $(l_2,l_3,l_4)$. The angle of $GH_h$ would be the angle component of the individual LH, which the $GH_h$ consists of. Hence the angle of $GH_h$ is the angle corresponding to one of $l_1$, $l_2$ or $l_3$.

In a similar way we find the angle for the other GHs in the system. The timestamps of $GH_h$ correspond to the timestamp of the points in the various LHS of $GH_h$. We say that $GH_h$ and $GH_l$ are in ascending manner of timestamp if the timestamp of $GH_h$ is before that of $GH_l$.

With these definitions, we proceed to compute the entire GHs for the target node in the system.

Once the nodes receive the GHs from its neighboring nodes (lines 5,6,7), it checks if the list of GHs received and its own list of GHs satisfy the criteria mentioned in the above section (line
8). If they’re satisfied, then it merges the two list of GHs (line 9) and sends the summarized list of GHs to its neighboring node at a lower level. It sends this list to all the neighboring nodes which lie at a lower level (lines 11, 12, 13).

If we assume that the list of GHs obtained from the higher level node is $GH_{higher} = (h_1, h_2, h_3)$ and its own GH is $GH_{lower} = (h_3, h_4, h_5)$, it checks if $\text{Angle}(GH_{higher}) = \text{Angle}(GH_{lower})$; $\text{Timestamp}(GH_{higher})$, $\text{Timestamp}(GH_{lower})$ are in ascending manner.

If these are satisfied, the resultant GH by merging the two GHs ($GH_{higher}$ and $GH_{lower}$) is $GH_{resultant} = (h_1, h_2, h_3, h_4, h_5)$

If the criteria are not satisfied (line 10), the node simply discards the list of GHs it received and sends only its list of GHs to all the neighboring nodes at a lower level. The above steps are performed by all the nodes at all the levels in the system.

In a more generic way, we can say that a node, upon getting GHs from its neighboring node, combines them with its own GHs if the criteria are satisfied and then sends the resultant consolidated GHs to all its neighbor(s) at a lower level. In this way, we can envision a flow of messages from the outside or the periphery of the network to the center of the system which is represented by the target node. This exchange of messages is done till the consolidated GHs reach the target node (line 15, 16). Since the target node is at level zero and all the other nodes are at a level greater than zero, we can safely say that all the nodes in the system have been covered. The target node gets the consolidated list of GHs from its neighbors and combines them based on the criteria to form the final, summarized and consolidated Global Hypotheses which is the desired result. Thus, a target node is able to compute the Global Hypotheses based on the message flow from the outer level of nodes towards the inner level of nodes. Fig 3.3 gives a vivid picture of the flow of messages.
One of the main differences in this method when compared to the previous section is that only the target node has information about its complete and summarized GH. Among the other nodes in the system, it is possible that some of the nodes have a partial knowledge of their GHs and the others have no knowledge of their GHs. There are many ways to place the target node in the system. It can be placed at the center, edge or the corner of the system. Depending on the position, the number of messages required to compute the GHs change. More details about this are explained in Chapter 7. In any case, the total number of messages required is much lesser than the number required for computing the GHs for all the nodes in the system. As we see in Chapter 6, the amount of messages saved is about 67% which is a very big saving.

We now present an example scenario which explains how to go about computing the LHs for nodes in the system as well as covering both the cases: computing of GHs for the target node as well as all the nodes in the system.
Chapter 5

Example Scenario

5.1 Introduction

In this chapter, we illustrate the steps of our algorithm with an appropriate example. The first section of this chapter explains the computation of LHs for all nodes in the system. The next section illustrates the computation of GHs of length three for all nodes. Next, we illustrate both possible scenarios for further computation of GHs: We initially assume that the GHs for all nodes in the system have to computed. In the end, we describe the process of computing the GHs for a target node only.

5.2 Computation Of Local Hypotheses

The first step of our algorithm is to compute the list of Local Hypotheses for all sensor nodes in the system. The following criteria are checked before the LHs are generated for each of the nodes:

1). Taking sets of three points or more, check whether they lie on the same line in the same
2). Check whether the points are in ascending/descending timestamp fashion.

In Fig 5.1, we assume a 10 X 10 system with a total of 100 nodes. We see the list of points noted by each sensor node in the system. These need not necessarily represent all the points for all the nodes. Sufficient points have been taken to illustrate our algorithm. The points marked with the rectangular box are those which satisfy the criteria for forming LHs. Hence these points form an LH for that particular node. We have shown three different LHs (LH1, LH2 and LH3) in Fig 5.1. There are other LHs in other nodes which can also be determined. Note the set of points at the top. These are in the same line and are in ascending timestamp fashion. But they are only two in number and since our algorithm requires a minimum of three points to form an LH, these two do not form an LH for that node.
5.3 Computation Of Global Hypotheses Of Length Three

This is the second step of the algorithm in which Global Hypotheses of length three are calculated. This step involves merging of LHs from neighboring nodes based on whether they satisfy the criteria or not. Only one iteration is performed and we get the GHs of length three as the result. From Fig 5.2, we can see that there are three sets of GHs formed (GH1, GH2 and GH3). We take LHs from neighboring nodes for all nodes and merge them to obtain the GHs. There are many more GHs formed as can be seen from the figure. To determine which node the GH belongs to, we find the node number of the central node in the GH pattern. The GHs shown correspond to different nodes; GH1 - node five, GH2 - node fortyfive and GH3 - node sixty. Note the nodes in the Fig 5.2 towards the upper left part. These nodes are not a part of any Global Hypothesis because they do not have any neighboring nodes to contribute to the GH. As we've mentioned before, there has to be at least three nodes to form a GH. In this way, we compute the GHs of length three for all nodes.

Figure 5.2: List of GHs of length three for Sensor Nodes in a 10 X 10 System
The above two steps are common for both the variants of our algorithm. The next steps are chosen based on whether we want to compute the GHs for all the nodes in the system or only for the target node.

### 5.4 Computation Of Consolidated GHs For All Nodes

![Figure 5.3: List of GHs of length five for Sensor Nodes in a 10 X 10 System](image)

This step results in the consolidated list of summarized GHs for all nodes in the system. In this step, we compare the GHs obtained from the previous step of neighboring nodes. If they satisfy the criteria, we merge them to form a consolidated GH of an increased length. For example in Fig 5.3, we see that for node fortyfive, the length of the GH has increased from three (in the previous step) to five. Now the GH involves five nodes instead of three. With every iteration, the length of the GH or the number of nodes covered increases by either one or two depending on the location of the node. We see from Fig 5.3 that the length of all the three GHs shown in Fig 5.2 have increased from three to five. This is the result of one iteration. If we repeat the iterations till we reach the end of the network, the result would look like in Fig
Here we can see that the three GHs (GH1, GH2 and GH3) encompass all the possible nodes in the system which satisfy the criteria for that particular GH. We say that GH2 has covered all the relevant nodes in the system and is the resultant summarized global hypothesis for node fortyfive. Similarly, the consolidated GHs for nodes five and sixty are GH1 and GH3 respectively.

All nodes which form a part of a GH own that particular GH. In other words from Fig 5.4, we can say GH2 is a possible consolidated GH for nodes 1, 12, 23, 34, 56, 67, 78, 89 and 100. It is possible for a node to have or be a part of more than one GH as is shown by nodes 1, 10 and 100 in Fig 5.4. It is also possible for the length of a GH to be less than the system length. For example in Fig 5.4, the length of GH4 is 5 whereas the length of the system is 10 (Assuming it is a 10 X 10 system).

Also, it is possible to manipulate the length of the GHs formed by limiting the number of
iterations performed. For example in Fig 5.3, the GHs formed are of length five and is the result at the end of one iteration. If we are interested in computing the GHs of length seven for all the nodes in the system, then we perform two iterations.

In this way, we compute the consolidated summarized Global Patterns/Hypotheses for all the nodes in the system.

5.5 Computation Of Consolidated GHs For Target Node

![Figure 5.5: List of GHs of length five for Target Node in a 10 X 10 System](image)

In this step, we assume that the target node is already known by all nodes in the system. Also the GHs of length three are already formed. The next step is to form the GHs of length five. The nodes in the higher level send messages (containing the list of their GHs) to neighboring nodes in the lower level. The direction of flow is inwards into the system towards the target node as shown Fig 5.5. The GHs are formed in a similar way as in the previous step. GHs of the two nodes are checked for the criteria. If they satisfy, they are merged to form a GH of length five which starts from the node at the highest level and ends with the node at
the lowest level as shown in Fig 5.5. GH1 and GH2 are shown in the figure. In this case, there are four possible GHs possible in Fig 5.5. Note that in this variant of our algorithm, the GH always starts from the node in the outer most level and progressively advances towards the inner network whereas in the previous variant, a GH could be formed at any part of the network; at the center or at the periphery. Also in this variant, the GH cannot contain nodes all of the same level.

![Diagram of GHs in a 10 X 10 System](image)

**Figure 5.6: List of complete GHs for Target Node in a 10 X 10 System**

Once the GHs of length five are formed, the nodes which performed the merging, now send this list of GH to their inner neighboring nodes as shown in Fig 5.6. After checking and merging of GHs, a new consolidated GH is formed which is of a greater length. This procedure continues till the list of GHs reach the target node. At the target node, the GHs obtained from its neighboring nodes from all directions are merged into one or more GH based on the criteria and are shown in Fig 5.6. We can see from the figure that GH1 and GH2 are the two GHs for the target node. This is the final result of our algorithm. The GHs for the other nodes are not known. For example in Fig 5.6, the GH for node 5 is not known because node 5 was not the target node.
We say that nodes 1, 12, 23, 34, 45, 67, 78, 89 and 100 have a partial knowledge of their GHs. From Fig 5.6, we can say that these nodes are a part of GH2 and hence have GH2 as a Global Hypothesis. But it is possible for them to own other GHs which is unknown because they were never assigned as target nodes. The entire list of GHs is known only for the target node.

In this way, we compute the list of the consolidated Global Patterns for either the target node or all the nodes in the system.
Chapter 6

Complexity Analysis

6.1 Introduction

In this section, we discuss the complexity of our algorithm. In sensor networks, bandwidth is of a lot of importance and is always limited. The goal in such a system to minimize the bandwidth used by the sensors in the system. We apply this same idea to the complexity analysis of our algorithm. We define the complexity of our algorithm in terms of the number of messages exchanged between the nodes in the system and always try to minimize it.

We consider two cases for analysis of complexity. The first corresponds to the first variant of our algorithm where in we compute the Global Hypotheses for all the nodes in the system. In the second, we consider a target node and compute the complexity for computing the GHs for that target node only. However, for both these cases, the initial couple of steps are the computing of the Local Hypotheses and the Global Hypotheses of length three. They are common for both the variants of our algorithm and hence, we first calculate the complexity of these two steps. Then we consider the two cases and arrive at an expression for the complexity for each of them.
6.2 Computation Of LHs

The computation of LHs for the sensor nodes is performed at the local data site. Hence there is no exchange of messages at this step because of which the complexity of this step is negligible. In the next step, every node communicates with each of its neighbors to compute its list of GHs.

6.3 Computing GHs Of Length Three

For the computation of GHs, each node gets the list of LHs from all its neighbors. From the placement of the nodes in the system, we know the number of neighbors for every node; this ranges from three to eight. A node at the corner has three neighbors, where as a node on the edge has five neighbors and a node anywhere else has eight neighbors as is shown in Fig 6.1.

If we take into account a message exchanged for every neighbor for every node in the system, we need to find the total number of neighboring nodes for all the nodes in the system to arrive at a number for the total number of messages exchanged. We calculate the total number of neighbors as follows:

We assume a 10 X 10 system in which there are a total of 100 nodes placed in a square grid as shown in Fig 6.1. As can be seen from Fig 6.1, there are 4 nodes in the corner, 32 nodes on all four edges and the remaining nodes elsewhere are 64. Also, we can see from the figure that

For a node at the corner, \(N_{\text{corner}}\), the number of neighbors = 3;

For a node on the edge, \(N_{\text{edge}}\), the number of neighbors = 5;

For a node elsewhere, \(N_{\text{others}}\), the number of neighbors = 8.
If we apply the above numbers, we calculate the total number of neighbors:

for all nodes at the corner = 4 * 3 = 12;
for all nodes on the edge = 32 * 5 = 160;
for all other nodes = 64 * 8 = 512.

The total of the number of neighbors for all nodes in the system is = 12 + 160 + 512 = 684.
Hence the total number of messages exchanged in a 10 X 10 system for computing the GHs of length three for all the nodes is 684.

In a more generic case, we consider an n X n system in which we have \( n^2 \) nodes and the size of the system is n.
The number of nodes at the corner in such a system remain the same at 4.
The number of nodes on the four edges is = 4(n-2).
The number of nodes elsewhere in the system = \((n - 2)^2\).

The computing of total number of neighbors for all nodes in the system is as follows:
Total number of neighbors for nodes at the corner: \(3 \times 4 = 12\).

Total number of neighbors for nodes on the four edges: \(5 \times 4(n - 2) = 20(n - 2)\).

Total number of neighbors for nodes elsewhere in the system: \(8 \times (n - 2)^2\).

Hence the total number of neighbors for all the nodes in the system

\[= 12 + 20(n - 2) + 8((n - 2)^2) = 8n^2 - 12n + 4.\]

The next step in the analysis of the complexity is the computation of the GHs. We consider both the variants of our algorithm.

### 6.4 Computing Entire GHs For All Nodes

In this step, we compute the entire GH for all nodes in the system. Messages are exchanged between nodes and all their neighboring nodes. However, the messages exchanged are the GHs (GHs of length three formed at the end of the previous step) present at each node. In the previous step, these messages were the LHs present at each of the nodes. The number of messages needed for this would be the same as before number of messages exchanged = \(8n^2 - 12n + 4\).

At the end of the first iteration, all nodes complete the exchange of messages with all their neighbors to form a GH of a greater length. However it is possible for a node, that the GHs do not exist in which case, the original GH of length three will remain at that node. If there are new GHs formed, then their length increases by either one or two depending on their location in the system. For every new GH formed, there is either a single node or a couple of nodes included in that new GH depending on the position of the node. If a node is on the edge or at the corner of a system, the length of the new GH increases by one. If the node is anywhere else in the system, the length increases by two. For this same reason, a node in the center of...
the system forms its entire GH much more quickly than a node on the edge or at the corner of a system.

As we discussed before, it is possible that the entire GH may not be formed for nodes at the end of the first iteration. If we assume that nodes do form a part of GHs which are of a length equal to the length of the system (n X n), we need to repeat these iterations to get those entire GHs for all the nodes. Again, the number of iterations required to compute the entire GHs for a node depends on the position of the node. If we assume the system to be an n X n system, then

- if the node is placed on the edge or at the corner of the system, the number of iterations required to form a complete GH is n-1

- if the node is anywhere else in the system, it requires less than n-1 iterations. The number decreases as the position of the node goes towards the center of the system. At the center, a node requires as less as (n/2 - 1) iterations.

If we assume that all nodes computed new GHs of increased length in every iteration, then a very interesting point here is that at the end of every iteration, every node replaces its existing list of GHs with the new GHs which was formed during that iteration. It uses these updated GHs as messages to exchange with the neighboring nodes during the next iteration.

In this way, the messages (or the updated GHs) keep getting exchanged between the nodes to eventually form a global summarized GH. As discussed before, after every iteration, the length of the GH increases by one or two. The maximum length of such a GH would be n nodes where n is the size of the system.

If we require k such iterations to form the complete GHs, the total number of messages exchanged would be k * (8n^2 - 12n + 4).

Hence the complexity of the system is k * (8n^2 - 12n + 4). We now define the best case and
the worst case scenarios for this system.

**Best Case Scenario:**
We define the best case scenario as the case in which it takes the least number of iterations to form the entire summarized Global Patterns. This occurs for the nodes in and around the center of the system. Since they have neighboring nodes on both sides to contribute to the growth of their GHs, the length of their GHs increase by two during every iteration. Thus it takes only $n/2 - 1$ iterations to form the entire GHs for these nodes, assuming that its an $n \times n$ sized system. For the best case, $k = n/2 - 1$ and hence the complexity is

$$(n/2 - 1) * (8n^2 - 12n + 4) = 4n^3 - 14n^2 + 14n - 14$$

**Worst Case Scenario:**
We define the worst case scenario as 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 nodes on only one side to contribute to the growth of their GHs, the length of their GHs increase by only one during every iteration and thus it takes $n - 1$ iterations to form the entire GH, assuming that its an $n \times n$ sized system. For the worst case, $k=n-1$ and hence the complexity is

$$(n-1) * (8n^2 - 12n + 4) = 8n^3 - 20n^2 + 16n - 4$$

Hence we can say that the complexity is of the order $O(n^3)$

**6.5 Computing GHs For Target Node Only**

In this case, the user specifies a pre-defined target node from among the nodes in the system. The GHs are then computed for that node only. The first step in this case is the target node informing all the nodes in the system about itself being the target node. This is done through exchange of messages which add to the complexity of this case. The total number of messages
required to do this in a worst case scenario is 684 (for a 10 X 10 system) 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. This is a sequential step where in the assigning of levels to all the nodes start with level 0 or the target node and go on till all the nodes have been assigned a level. The number of messages required to do this is 684.

Next, messages start getting exchanged from nodes in the outer level to those in the inner level. In this way, we ensure that the the number of messages exchanged is brought down to a bare minimum because we make sure that nodes do not exchange will all their neighbors. They exchange only with those neighbors which are at a lower level.

If we assume a 10 X 10 system, with the target node placed at the center of the system, then the number of messages exchanged to form the entire GHs for the target node is 224. Hence the total number of messages required to form the entire GHs for the target node is = 224 + 684 + 684 + 684 = 2276. And the total number of messages required to form the GHs for all the nodes in a 10 X 10 system = 6156 (as is shown in the next Chapter).

Hence we see that for the second case, the number of messages is about 33% of the number of messages in the first case which means that we save about 2/3 the number of messages or the communication costs in the network.

In this way, we reduce the complexity of our algorithm by varying the requirements, slightly. The trade off with this method is that the complexity of the algorithm decreases, but the amount of knowledge available also decreases. In the first case, all the nodes knew about their entire GHs; in the second case, only the target node knows about its entire GH. The other nodes have either a partial knowledge or no knowledge of their GHs. Hence the amount of
information known goes down in the second case.

If we assume the average number of levels to be $l$, average number of nodes in each level to be $n_l$ and the average number of neighboring nodes with a lower level for every node to be $n_g$, then the total number of messages exchanged is $l(n_l)(n_g)$. It turns out that this number is negligible when compared to the number of messages required for computing the GHs for all the nodes in the previous case.

Hence the total number of messages required to compute the GHs for the target node is $(8n^2 - 12n + 4) + l(n_l)(n_g)$ which effectively is of the order $O(n^2)$.

In the experiments done, the number of messages required for the target node were found to be about 33% of those required for all the nodes. Hence we see that there is significant amount of savings as far as complexity of the system is concerned.

We now present a comparative study between the two variants of our algorithm.

### 6.6 Points Of Comparison Between The Two Variants

- Both the variants compute the consolidated summarized Global Patterns for sensor nodes in the system.
- We perform in-network aggregation in both the the variants of our algorithm.
- The flow of messages is from a node to its neighboring node only. This is true for both the variants.
- Intelligent sensor nodes capable of capturing geographical information and scalar variables like temperature, motion, sound, etc are used in both the variants of our algorithm.
• We place the sensors in a square grid for testing in both our variants. Also we assume that the sensing range for any pair of adjacent sensors is mutually exclusive of each other for both the variants.

• The first two steps of computing the LHs and GHs of length three are the same for both the variants.

• The criteria for forming the LHs and the GHs are the same for both the variants of our algorithm. Also the list of neighbors remain the same for each of the nodes.

6.7 Points Of Contrast Between The Two Variants

• The first variant computes the GHs of all the nodes in the system whereas the second variant will compute the GHs 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.

• The first variant results in all nodes knowing about their complete list of GHs whilst the second variant results in all the nodes knowing a partial or no list of their GHs.

• 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.

• 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.

- 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 GH formed in the second variant, no two nodes can be at the same level. The level for each node in the GH has to be different from all the other nodes in the system; this does not matter for the GH formed in the first variant.

- 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. Again, we assume here that we are talking about those nodes only which form a GH of which the target node is a part.

- 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.
Chapter 7

Results

In the previous sections, we illustrated our algorithm with suitable examples. We now present the results of the tests we had run for both the variants of our algorithm. We developed our algorithm in Java.

We refer to messages as the list of LHs and GHs. Also we refer to the two variants of our algorithm as Case 1 and Case 2.

We choose the following metrics to generate results for and to compare the two cases:

- Total number of nodes in system
- Location of target node in the system (This will be applicable to Case 2 only)
- Total number of iterations (This will be applicable to Case 1 only)

The results for all the test cases run for both the variants of our algorithm is as follows:
7.1 Total Number Of Nodes In System

The nodes in the system are distributed in a grid manner (the shape of the grid is assumed to be a square). Also, the sensing range of every node is mutually exclusive of the sensing range of any of its neighboring node. For example, if we have a grid of \( n^2 \) nodes, then the size (or the length) of the grid will be \( n \). In this section, we varied the number of nodes in the system from 25 to 100 (from a 5 X 5 grid to a 10 X 10 grid) and computed the number of messages exchanged between nodes for both variants of our algorithm.

![Figure 7.1: List of GHs for Sensor Nodes in a 5 X 5 System](image)

**Case 1: GHs computed for all nodes in the system:**

The number of messages exchanged between nodes in the system increase with increase in the size of the grid. The grid size was varied from 5 X 5 to 6 X 6 and so on till 10 X 10 (a couple of grid sizes 5 and 7 have been shown in Fig 7.1 and Fig 7.2) and the obtained results are shown in Table 7.1. The increase in size of the grid from 5 X 5 to 6 X 6 results in a saving of about 50% of messages whereas a 10 X 10 size results in a saving of about 67%.
Case 2: GHs computed for target node only:
In this case, we fixed the target node as is shown in Fig 7.3 and varied the size of the grid from 5 X 5 to 10 X 10 (a couple of grid sizes—5,7 are shown in Fig 7.4 and 7.5). Both the figures show the levels for each grid. In this case, we define the number of messages exchanged to be the sum of the number of messages required to form the levels and the number of messages required to form the final consolidated GHs. As is expected the number of messages increase with an increase in the grid size. The increase in each case is by almost 50%.

7.2 Location Of Target Node In The System
In this section, we compare the number of messages exchanged with different positioning of the target node in the system. This section is not applicable to Case 1 as the concept of a target node does not apply to it. We assume that the size of the grid is 10 X 10 with the total number of nodes in the system to be 100. Also, we assume that the Global Hypotheses of length three have already been calculated and we do not include those messages in the total count. Next,
Figure 7.3: List of GHs for Sensor Nodes in a 10 X 10 System

we vary the position of the target node from the center of the grid (Fig 7.6) to the edge of the grid (Fig 7.7) and finally to the corner of the grid (Fig 7.8).

The number of levels formed is lesser when the target node is placed at the center than at the edges or at the corner. From our algorithm, we deduce that the number of messages increase with the number of levels in the system. Hence the complexity or the number of messages increase when the target node is placed at the edge or at the corner of the system. The number of levels in Fig 7.7 (target node at the edge) and in Fig 7.8 (target node at the corner) is more than that in Fig 7.6 (target node at the center). In Fig 7.6, we have 5 levels where as in Fig 7.7 and Fig 7.8, we have 9 levels. These results are tabulated in Table 7.2.

Also, the other results obtained in this section are shown in Table 7.2. The number of messages, when the target node is at the center, is 224 and is less than the number of messages when the target node is at the edge or at the corner (the number is 244). Interestingly, the number of messages required for the last two cases is almost the same (243 and 244).
7.3 Total Number Of Iterations

We refer to an iteration as a procedure in which all the nodes obtain GHs from their neighboring nodes and merge them with their own GHs based on the criteria to form a list of consolidated GHs. The length of GH increases with every iteration. Note that this iteration starts after all the nodes in the system have knowledge of their own GHs of length three. This section does not apply to Case 2 as, only a single iteration is required to compute the list of GHs. But in Case 1, we require more than one iteration to obtain the entire list of summarized GHs. The number of messages required to form Global Hypotheses of length three is 684 (for a 10 X 10 system). We do not include these messages in our results.

In this section, we vary the number of iterations from one to eight. The number of messages increase with increase in the number of iterations. The results for each iteration are tabulated in Table 7.3. Apart from the messages, the amount of information available to each of the nodes also increase with the number of iterations. With every iteration the number of
nodes, that form a part of GH, increase. This result is also shown in Table 7.3. We assume that
the node in consideration is a node on the edge or at the corner. We also provide the results
for a node located anywhere else in the system in Table 7.4. Even in this case, the number of
messages increase with increase in the number of iterations. The only difference is that the
length of the GH for these nodes increase by two whilst they increase by one in the former. It
takes just 4 iterations to form the entire GH for the node whilst it was 8 in the former. 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 (gridsize/2
- 1) iterations.

Also, the number of neighbors for any node is dependent on the position of the node. It
was observed that the number of neighbors for a node on the edge is five where as for a corner
node it was three. All other nodes in the system had as many as nine neighbors. Since a node
on the edge and the corner has a lesser number of neighboring nodes than any other node in
the system, it takes a higher number of iterations to compute its entire GH. From Fig 7.2,
we see that a node at the corner or at the edge has neighbors on one side only whereas other nodes have neighbors on both the sides. These neighbors contribute to the GH from both the ends whereas for the nodes on the edge and the corner, their GHs are being built from one side only. Hence these nodes need about double the number of iterations to build their entire GH when compared to the other nodes in the system. This is the reason for the nodes on the edge/corners taking more iterations to complete their entire list of GH. The complete list of details is tabulated in Table 7.5.
Figure 7.7: List of Levels with Target Node Placed on the Edge of the System

Figure 7.8: List of Levels with Target Node Placed at the Corner of the System
Table 7.1: Comparison of number of messages exchanged with the number of nodes

<table>
<thead>
<tr>
<th>Size of system</th>
<th>No of nodes</th>
<th>No of messages – Case1</th>
<th>No messages – Case2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5X5</td>
<td>25</td>
<td>576</td>
<td>328</td>
</tr>
<tr>
<td>6X6</td>
<td>36</td>
<td>1100</td>
<td>515</td>
</tr>
<tr>
<td>7X7</td>
<td>49</td>
<td>1872</td>
<td>724</td>
</tr>
<tr>
<td>8X8</td>
<td>64</td>
<td>2940</td>
<td>975</td>
</tr>
<tr>
<td>9X9</td>
<td>81</td>
<td>4328</td>
<td>1260</td>
</tr>
<tr>
<td>10X10</td>
<td>100</td>
<td>6156</td>
<td>1592</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison of number of messages exchanged with the location of the Target Node

<table>
<thead>
<tr>
<th>Position of Target Node</th>
<th>No of messages exchanged</th>
<th>No of levels formed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center of Grid</td>
<td>224</td>
<td>5</td>
</tr>
<tr>
<td>Edge of Grid</td>
<td>243</td>
<td>9</td>
</tr>
<tr>
<td>Corner of Grid</td>
<td>244</td>
<td>9</td>
</tr>
</tbody>
</table>
Table 7.3: Comparison of number of messages exchanged with the number of Iterations performed - for a node on the edge/corner

<table>
<thead>
<tr>
<th>No of Iterations</th>
<th>No of messages exchanged</th>
<th>No of nodes in GH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1368</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2052</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2736</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3420</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4104</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>4788</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>5472</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>6156</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 7.4: Comparison of number of messages exchanged with the number of Iterations performed - for a node in the system apart from the edge/corner

<table>
<thead>
<tr>
<th>No of Iterations</th>
<th>No of messages exchanged</th>
<th>No of nodes in GH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1368</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2052</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>2736</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>3420</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 7.5: Comparison of number of Iterations required to form the Entire GH with the position of the node in a 10 X 10 Grid

<table>
<thead>
<tr>
<th>Position of Node</th>
<th>No of iterations for Entire GH</th>
<th>No of neighbors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge of System</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>Corner of System</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Anywhere else in the System</td>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>
Chapter 8

Conclusions

8.1 Summary

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 square grid and our algorithm was used to predict the trajectory of the moving object. We considered this equivalent to mining of global spatio-temporal 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 site 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.

Some possible applications of our algorithm are the following:

- An area can be under supervision for any possible movement of trucks or vehicles sensed by a sensor. Every sensor can act as a database and therefore a grid of sensors can act as distributed databases.

- A population of sensors can be used to monitor a terrain containing some chemical pollutant. Our algorithm can be used to monitor the flow of the pollutant in the soil or in the air.

We now present a list of possible areas of future work.

### 8.2 Future Research

A list of possible topics for future research include the following:

We considered a square grid for purpose of computing of Global Patterns for nodes in the system. There is scope for work in tuning our algorithm to make it work for a rectangular grid or any other shape like a circle, etc. Also, we assumed that the sensing range of nodes in the system is mutually exclusive of each other. But in real world cases, there is a certain amount of noise present in the sensing range of nodes due to the neighboring nodes. Incorporating the effect of noise on recording the details of the moving object for a node and designing an efficient way of doing it can be considered as possible future research work. The battery life
of sensors is short lived and it is possible to have an alternate source of power. This could be another battery or any other source of energy. This would be a very interesting topic of research.

One of the major assumptions in our work is that we consider linear GHs only. If the moving object were to change direction, it would not be possible for us to tell through our algorithm that the object changed its course of direction. This is another area of work wherein computation of non-linear GHs of objects are considered. A possible solution to this could be the assigning of an ID to every object and then detecting its motion in the network.

Also, we take into account a single object at any given instant of time. A possible area of work could include detection and futuristic prediction of movement of frontiers taken as a whole. A frontier could include a heat wave, ocean current, an earthquake, etc.
Bibliography


[22] 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


[33] Zhang D., Gunopulos D., Vassilis J. T and Bernhard S., Temporal Aggregation over Data Streams Using Multiple Granularities, *In Proceedings of Extending Database Technology*; pages 646-663, Prague, Czech Republic, 2002