I, Abhishek D Sharma, hereby submit this original work as part of the requirements for the degree of:

Master of Science
in Computer Science

It is entitled:
Discovery of Trajectory Clusters in Spatio-Temporal Data

Student Signature: Abhishek D Sharma

This work and its defense approved by:

Committee Chair: Raj Bhatnagar, PhD

Raj Bhatnagar, PhD
DISCOVERY OF TRAJECTORY CLUSTERS IN SPATIO-TEMPORAL DATA

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 (M.S.)
in the Department of Computer Science

10 August, 2010

by

Abhishek D Sharma

B.E., Computer Science and Engineering, University of Madras, 2003

Thesis Advisor and Committee Chair:

Dr. Raj Bhatnagar
Abstract

Extracting and Clustering of trajectories from Spatio-Temporal data is a challenging problem due to the highly exponential nature of the space of possible clusters. It is an important problem as it holds potential for obtaining insights into vast amounts of spatio-temporal data. Most of the existing algorithms for mining spatio-temporal data focus on clustering the sets of meaningful trajectories that are already identified and available to them. Discovering interesting trajectory clusters in a three dimensional spatio-temporal dataset in which each cell has been instantiated to a value is an extremely large problem due to the number of potential trajectories.

In this thesis we present an algorithm which uses divide-and-conquer strategy by finding clusters in few layers at a time and then combining these results to construct larger clusters. We demonstrate how this strategy emulates results that may be obtained for the complete datasets.
To my Brother and Sister in Law
Acknowledgments

I wish to thank my advisor Dr. Raj Bhatnagar for providing valuable guidance during the course of this work.

I thank my thesis committee members Dr. George Purdy and Dr. John Schlipf for their suggestions on this thesis work.

Most importantly, I would like to thank my family for the love and support they provided at every step and without whom I would not have achieved this.
# Table of Contents

1. Introduction  
   1.1 Motivation  
   1.2 Pattern Representation  
2. Research Aim  
3. Problem Definition  
   3.1 Formal Description  
4. Features  
5. Distance Functions  
   5.1 Distance  
   5.2 Importance of $\alpha$  
   5.3 Importance of $\beta$  
   5.4 Importance of $\gamma$  
6. Approach and Methodology  
7. Algorithm  
8. Proof  
9. Complexity Analysis  
10. Related Research  
11. Results  
12. References
List of Figures

1.1 Small Dataset
1.2 Close and Distant Trajectories
1.3 Narrowing Patterns
1.4 Broadening Patterns
1.5 Forking Patterns
1.6 Merging Patterns
1.7 Intersecting Patterns
1.8 More Specific Patterns
1.9 More Generalized Patterns
1.10 Two Patterns-Same Trajectories
1.11 Mirroring Patterns
1.12 2-D ↔ 3-D view of a dataset
1.13 3-D ↔ 2-D view of a dataset

3.1 Possible Directions from one cell
3.2 Exponential Increase in the number of trajectories
3.3 Trajectory of a simple cluster

4.1 Values of cells in a grid
4.2 Spatial Specifications of cells
4.3 Directions

5.1 Distance between two strings
5.2 Distance between a string and a cluster
5.3 Distance between two clusters

6.1 Nine Possible Directions
6.2 Nine Possible directions from a cell
6.3 Exponential increase in the number of trajectories
6.4 3-Length trajectories → 4-Length trajectories
6.5 Top-Down Approach
6.6 Bottom-Up Approach

11.1 Output-Close and Distant Patterns
11.2 Output-Pattern of varying sizes
11.3 Output-Overlapping Patterns
11.4 Output-Pattern with branches
11.5 Output-Intertwined Patterns
11.6 Output-Periodic Patterns
11.7 Output-Dense and Sparse Patterns
11.8 Input-Grid 1
11.9 Input-Grid 2
11.10 Input-Grid 3
11.11 Input-Grid 4
11.12 Output-$\alpha=0.4, \beta=0.4, \gamma=0.2, \theta=0.08, \text{Length} \geq 3$
11.13 Output-$\alpha=0.4, \beta=0.4, \gamma=0.2, \theta=0.08, \text{Length}=4$
11.14 Output-$\alpha=0.2, \beta=0.3, \gamma=0.5, \theta=0.08, \text{Length} \geq 3$
11.15 Output-$\alpha=0.2, \beta=0.3, \gamma=0.5, \theta=0.08, \text{Length}=4$
11.16 Output-$\alpha=0.3, \beta=0.6, \gamma=0.1, \theta=0.08, \text{Length} \geq 3$
11.17 Output-$\alpha=0.3, \beta=0.6, \gamma=0.1, \theta=0.08, \text{Length}=4$
Chapter 1

Introduction

Spatio-Temporal Databases contain regions or patterns with similar characteristics. Such Patterns include Spatio-Temporal patterns as well. Spatio-temporal patterns or trajectories are those which change in space with respect to time. These spatio-temporal patterns are obtained when a phenomenon with a spatial component changes over a period of time. Mining of such patterns enables us to understand the behavior of the phenomenon and predict its occurrence under suitable conditions. Here we propose a general algorithm for mining of such patterns which can be applied to many applications. For example, mining for Spatio-Temporal patterns on the weather data of a region, collected over a few months, can help us in finding those parts of the terrain which share the same pattern in a given time-window and in predicting the behavior over the next few time periods.

In the last few years the study of spatio-temporal patterns has gained considerable attention because of the massive amounts of accumulated data. Spatio-temporal data is represented as the values of the spatial attributes of the phenomenon under study at each time instant (or over a time interval). For example, for the weather data we represent the spatio-temporal data as weather parameters (Pressure, Temperature, Humidity, etc) in various parts of a city for each day, over a period of some years.
Mining for patterns on such databases would give us some very useful information which would be almost impossible to achieve manually because of the huge amount of data. We could, for instance, find out why during some particular months at some particular places in the city the weather follows a particular pattern. Knowing the reason for the occurrence of the phenomenon we can predict its behavior under a given set of conditions.

The Spatio-temporal data not only contains a spatial component, but a temporal component as well. Due to this the data is inevitably multidimensional, and this makes mining on such databases a very complex problem. Clustering on a 2 dimensional data-set is not a very difficult job. But adding one more dimension to the data-set increases its complexity exponentially. A sample of spatio-temporal dataset is shown in the figure 1.1.
Another problem which is closely associated with the mining of Spatio-Temporal patterns require clustering of Spatio-Temporal primitives, that is, the Spatio-Temporal trajectories that are already available to us and we need to cluster them on the basis of some similarity measure. These trajectories are often two dimensional with the first point of the trajectory defining the position of the object at time $t=t_0$, the second point defining the location of the same object at time $t=t_0 + T$, the third at $t=t_0 + 2T$ and so on. Thus, if we have $n$ objects moving in space over a period of time, we will have $n$ trajectories. The aim is to cluster these $n$ objects based on their movement patterns, that is, to cluster similar trajectories.
The many variations to this problem require each cluster to satisfy certain conditions. For example, the trajectories might be further apart but if they have a similar shape then they must belong to the same cluster (figure 1.2), or the orientation of a trajectory might be different but the shape may be similar. These different variations and requirements make this a difficult problem.

![Diagram of Clusters A and B with trajectories]

Fig 1.2: Close and distant trajectories.

Depending on our needs cluster $B$ may or may not be considered as a pattern and thus can be processed accordingly.

Spatio-temporal databases have received considerable attention during the last few years due to the accumulation of large amounts of multi-dimensional data that evolve
with time in industries such as mobile communication, traffic supervision and Geographic Information System (GIS) technology. While dealing with spatio-temporal databases and applications, one of the major issues is the representation of the phenomenon spatial attributes (which are very likely to change with time) with respect to time.

In our approach we divide the data into grids, each grid representing a temporal profile or a time stamp of the spatial region under consideration. Thus, if we have $t$ such profiles (at $n$ instances of time) we will have $t$ such grids or time stamps. Each grid is further divided into $x\times y$ cells where each cell represents a location within the particular spatial region. Thus, we have $t$ readings of $x\times y$ number of locations. The figure * below shows a small sample of such a spatio-temporal dataset where we have $t=5$ grids or time stamps and the spatial region consists of $4\times 10$ (i.e. $x\times y$ cells).
1.1 Motivation

In the last few years the study of spatio-temporal patterns has gained considerable attention because of the massive amounts of accumulated data. Spatio-temporal data is represented as the values of the spatial attributes of the phenomenon under study at each time instance (or over a time interval). For example: for the crime data we represent the spatio-temporal data as the number of crimes (spatial attribute) in the various parts of the city for each month, over a period of 3 years.

Mining for patterns on such databases would give us some very useful information which would be almost impossible to achieve manually because of the huge amount of data. We could, for instance, find out why during these particular months at these particular places in the city the crime rate follows a particular pattern. Knowing the reason for the occurrence of the phenomenon we can predict its behavior given a set of conditions.

Mining for Spatial patterns and temporal patterns has been an active field of study for quite some time now. Rakesh Agarwal and Ramakrishnan Srikant did a lot of work in the field of temporal patterns in the mid 90's and since then a number of algorithms have been developed to solve the temporal and spatial problems. But discovering patterns, which move both in space and time, has been difficult because of the computationally intensive nature of the problem.

We are trying to develop an algorithm which can extract relevant patterns from
the spatio-temporal database based on the distribution of data at each “time stamp”. A time stamp is a single time instance or a time interval for which data is collected. The collected data for a particular number of time stamps comprises of a spatio-temporal database.

The data is considered to be comprised of various grids where each grid represents the data with respect to each time stamp. The patterns which we want to retrieve may fall into different categories. Firstly, we are interested in those patterns which follow the same path down the grid (not necessarily through all the grids), that is, the patterns which move in the same directions over a period of time. We are also interested in patterns which move in opposite directions, that is, which complement each other. We also want to retrieve those patterns which do not change with time (spatial patterns). The figures below show some of the many possible patterns the user might be interested in.
The figure 1.3 shows that a cluster of trajectories originating from the vicinity of a few locations converge towards a specific and a much narrow location over a period of time, which may indicate a phenomenon which might force such a behavior in those trajectories. An example of such a phenomenon might be the frequency of crimes reported over a period of time in various areas, which might concentrate towards warmer locations as the weather becomes colder. The figure 1.4 shows opposite effect on the trajectories, i.e., spreading wider over time.
Fig 1.4: Broadening Patterns

Fig 1.5: Forking Patterns

The figure 1.5 above shows that a cluster of trajectories follow the same direction
for a while but diverge into separate clusters later on. The figure 1.6 might indicate exactly the opposite phenomenon, i.e., separate clusters converging into one after a while. These kind of patterns might reveal the phenomenon which may be forcing such divergence or convergence.

The figure 1.7 below shows two clusters of trajectories that intersect at some point of time but still retain their individual properties. This might indicate two exclusive phenomenons which affect the behavior of only one cluster and not the other. This can be of great interest if these phenomenon emerge from the same set of attributes or properties but are effective only within a certain range.
Figure 1.8 and Figure 1.9 below show two very interesting cases. In many cases, while clustering trajectories we might find that some trajectories branch out for a short while at some time instance. In most cases these branches may be much shorter than the actual length of the trajectories in the cluster. Now since these smaller branches or trajectories are a part of the main cluster, their significance solely depends on the need of the user and the knowledge he wants to extract from mining them. Depending on the necessities of the users these branches may be included in the patterns extracted, i.e., obtain more generalized patterns (figure 1.9), or can be discarded if the user is interested in very specific patterns (figure 1.8).
The criterion of including or discarding these branches in the final result adds another issue for the user and can be modified accordingly.

The dotted branch or trajectory in figure 1.8 indicates that the branch has not been included in the main pattern while the figure 1.9 indicates that the pattern has been included.
The figure 1.10 below shows another interesting case. Here we consider two separate clusters trajectories having the same orientation and following the same directions and paths, but are spatially distant from one another. Normally we would be inclined to include them in the same pattern if our emphasis on the spatial distance between any two patterns is not too high. But if their spacial attributes or properties do not belong in the same range (as we discussed about the intersecting trajectories) then we would have to consider them as two different patterns. The phenomenon responsible for such an effect would be interesting to look into.
Fig 1.10: Two patterns Same Trajectories

Fig 1.11 Mirroring Patterns
If we look at some more cases and datasets we would find many interesting instances where spatial attributes and phenomenon interact with each other to form various interesting patterns.
1.2 Pattern Representation

Due to the complexity involved in the three dimensional representation of the patterns we might sometimes represent the patterns diagrammatically in two dimension. To avoid any confusion or loss of information caused by the transformation from three dimensions to two dimensions, the conversion process is explained briefly.

Let us suppose that the data consists of $t$ time-stamps or grids. We first visualize each grid from the top view (line of sight parallel to the $T$ axis. This would give the $X-Y$ view (2-D) of each grid. Now we rotate the 2-D view such that all the three axis are aligned equally. This step is shown in the figure 1.12 below with 4 grids.

![2-D and 3-D view of a dataset](image)

Fig 1.12: 2-D ↔ 3-D View of a dataset
Now we align the grids such that the $X$-$Y$ plane is parallel to the line of sight and the $T$ axis is perpendicular to the line of sight. We also rotate the $X$-$Y$ such that all the trajectories are visible. The figure 1.13 below shows the transition from the 3-D view to the required 2-D view.

Fig 1.13: 3-D $\leftrightarrow$ 2-D View of a dataset

From here on we switch between the two views, depending on the case at hand.
Chapter 2

Research Aim

We are trying to develop an algorithm which can extract relevant patterns from the spatio-temporal database based on the distribution of data at each grid. A grid is a single time instance or a time interval for which data is collected. The collected data for a particular number of grids comprises of a spatio-temporal database. The data is considered to be comprised of various grids where each grid represents the data with respect to each grid. The patterns which we want to retrieve may fall into different categories. Firstly, we are interested in those patterns which follow the same path down the grid (not necessarily through all the grids), that is, the patterns which move in the same directions over a period of time. We are also interested in patterns which move in opposite directions, that is, which complement each other. We also want to retrieve those patterns which do not change with time (spatial patterns).
Chapter 3

Problem Definition

The dataset is divided into $T$ timestamps. Each grid is in the form of a 2-dimensional grid. Each grid consists of $X \times Y$ cells. We represent each cell as $v(x,y,t) = f_1, 2, \ldots, k$ where $1 \leq x \leq X$, $1 \leq y \leq Y$ and $1 \leq t \leq T$. $k$ is the total number of quanta into which the data is divided.
3.1 Formal Description

Our algorithm is developed in the context of grid-based datasets from domains having spatio-temporal characteristics. The grid is assumed to represent the spatial image of observations taken at some point in time. A number of spatial grids are stacked over each other, in order of time instants at which each grid is observed. For example, let us consider each grid to be covering a city and each cell of the grid representing a number of blocks in the city. The data stored in each cell may represent the number of crime incidents that occurred in a month in that cell of the city. Successive grids represent the spatial crime distribution for successive time instants.

**Temporal Patterns:** A string of cell-values connecting, say, cell (1, 1)’s, would represent the temporal characteristic of the crime incidents in cell (1, 1) of the city. If our grid is of size 10X10 then there are 100 temporal profiles, one for each cell in the grid. With these 100 temporal profiles we can use an algorithm for mining temporal patterns to discover frequently occurring temporal substrings or subsequence patterns in the dataset.

**Spatio-Temporal patterns:** Spatio-temporal patterns are characterized by substrings or subsequences that occur close together, and or frequently, across the three dimensions of space$(x,y)$ and time$(t)$. We characterize this by defining a more general set of strings in the context of the grid example shown in Figure *. Starting from a cell $(x, y)$ a string connects to any
one of the nine adjacent cells in the next grid as shown in the Figure 3.1 below. So, there are nine two-length strings and 81 three-length strings starting from any single cell of a grid as shown in Figure 3.2 below. The number of such possible strings explodes very quickly but it is this set of strings that contains the spatio-temporal patterns. It is not possible to make explicit this humongous set of strings and it must remain only implicitly specified for any mining algorithm that wants to be scalable and efficient. An example of a pattern discovered by such algorithms is of a crime spree that moves from one neighborhood to other as weather changes from summer to winter and then back to the older neighborhoods in the next summer. The methodology is applicable to other situations where the grids may represent various types of pollution, weather, or social data.

Fig 3.1: Possible directions from one cell.
Fig 3.2: Exponential increase in the number of trajectories.

Number of Grids = 2

Number of strings originating from
1 Cell in grid ‘n’ and continuing till
Grid ‘n+1’ = $1 \times 9 = 9$.

Number of Grids = 3

Number of strings originating from
1 Cell in grid ‘n’ and continuing till
Grid ‘n+2’ = $1 \times 9 \times 9 = 81$. 
Fig 3.3: Trajectory of a simple cluster

We assume a three dimensional space where X and Y are the spatial dimensions and T is the time dimension. Each cell \((x, y, t)\) in this space stores a value given as \(v(x, y, t)\) and selected from a set of possible values: \(\{1, 2, \ldots, k\}\), where \(1 \leq x \leq X_{\text{Max}}\), \(1 \leq y \leq Y_{\text{Max}}\), and \(1 \leq t \leq T_{\text{Max}}\) and \(k\) is the maximum number of possible integer values (possibly quantized real values) from which a cell may take one of the values.

A sequence \(s\) in this data set is constructed as follows. We start from a cell in the \(X\)-\(Y\) grid at time \(t\) and let us say its value is \(v_t\). We then move to the \(X\)-\(Y\) grid at time \(t+1\) and select another cell which can be any one of the nine immediate spatial neighbors of the original cell. Figure-xx above shows the nine immediate spatial neighbors of the cell marked “5”, including the cell itself. We pick the value of this cell as \(v_{t+1}\) and append it to \(v_t\). Continuing in this manner we can construct a \(k\)-length sequence: \((v_b, v_{t+1}, v_{t+2}, \ldots, v_{t+k-1})\). Further, we let \(S_t\) be a set of sequences starting at time instant \(t\), at any location in
X-Y grid, and continuing for different possible lengths. The \(i^{th}\) sequence of length \(n\) (\(n\)-sequence) in set \(S_i\) is represented as \(S_i(i)\). A sequence, any member of set \(S_i(I)\), can be represented in any one of the following two forms:

Let \(d_i\) be the direction in which the sequence moves from time point \(i-1\) to time point \(i\), (nine possible direction numbers are shown in Figure-xx above) and let \(1..k\) be the discrete values, one of which, \(q_i\) a cell can take. The first representation for a sequence starts by stating an \((x, y)\) location, a value in that cell \(q_i\), followed by a direction \(d\) for the next cell in the sequence, followed by the value in that cell and so on. In the second representation form a profile is a sequence of points containing complete \((x, y), t\), and \(q_i\) value for each point of the profile.

These forms, typically, look as follows:

1. \(<x_{it}, y_{it}, q_{it}>, <d_{it+1}, q_{it+1}>, \ldots, <d_{it+n-1}, q_{it+n-1}>\>

2. \(<x_{it}, y_{it}, q_{it}>, <x_{it+1}, y_{it+1}, q_{it+1}>, \ldots, <x_{it+n-1}, y_{it+n-1}, q_{it+n-1}>>\>

An interesting cluster \(C\) of sequences is a set of sequences \(<S_1, S_2, \ldots, S_m>>\) such that they are close to each other from the perspective of some distance functions. Our objective is to discover such interesting clusters from the set of all possible sequences that exist in the dataset.
Chapter 4

Features

There are three main features embedded in a sequence. The first is the spatial transitions (directions along which it turns), the second is the values at each time instant that it takes and then we have the spatial specification. There are three variables with which we define a string and subsequently a pattern. Firstly, there is the value associated with each cell, and this is the quantity associated with each string at a particular grid. Then have the spatial specifications of a cell with respect to the other cells on the same grid. The spatial specifications of two cells give us the spatial distance between them. Then we also have the direction the string takes from one grid to the other. We’ll discuss these three variables in detail.

Value ($v$)

The quantitative representation of each cell at a particular time is given by its value ($v$). This value of a particular cell may vary in different grids (time-stamps). These values also specify the temporal attributes or temporal properties of a cell within a particular grid or at a particular time-stamp. We use these values (between two cells) to calculate a part of the distance function. Figure 4.1 below shows the values of cells in two grids or two different time-stamps.
Fig 4.1: Values of cells in a grid \((v)\)

**Spatial Specifications \((x, y)\)**

The spatial attribute or spatial specification of a cell defines its location on a grid. The spatial position can also be used to determine the spatial distance between two cells on a grid. Here we are interested in the Euclidean Distance between any two cells on a grid and it is this distance that we use in our distance function.
Fig 4.2: Spatial specifications of cells

**Direction ($d$)**

The direction of the string defines the path a string takes between two consecutive grids. As shown above a string can travel in any one of the 9 directions from one grid to the next.
Fig 4.3: Directions
Chapter 5

Distance Function

Let \( <x_{it}, y_{it}> \) be the starting position of the sequence \( S_i \), let \( q_{it} \) be the value of the string \( S_i \) at grid \( t \) and let \( d_{it}^j \) be the direction the sequence takes going from grid \( t \) to grid \( t+1 \).

We define two sequences \( S_a \) and \( S_b \) of length 3 as:

\[
S_a \rightarrow <x_{at}, y_{at}, q_{at}>, <d_{at+1}, q_{at+1}>, <d_{at+2}, q_{at+2}>
\]

\[
S_b \rightarrow (<x_{bt}, y_{bt}, q_{bt}>, <d_{bt+1}, q_{bt+1}>, <d_{bt+2}, q_{bt+2}>
\]

Now, we define our three distance functions \( f_1, f_2 \) and \( f_3 \) as:

\[
f_1 = AVG. (Q_t + Q_{t+1} + Q_{t+2})
\]

\[
Q_t = |q_{at} - q_{bt}|
\]

\[
Q_{t+1} = |q_{at+1} - q_{bt+1}|
\]

\[
Q_{t+2} = |q_{at+2} - q_{bt+2}|
\]

\( q_{at} \) and \( q_{bt} \) are the values or magnitudes of cells \( a \) and \( b \) at the grid or the time-stamp \( t \). It is of importance to note that \( a \) and \( b \) need not be at the same location at time-stamps \( t, t+1 \) and \( t+2 \). This is because \( a \) and \( b \) do not represent a location of but those cell through which the trajectories \( a \) and \( b \) traverse from at \( t, t+1 \) and \( t+2 \). \( f_1 \) represents the average of the absolute differences between the temporal attributes at time stamps \( t, t+1 \)
and \( t+1 \).

\[
f_2 = AVG. ( D_t + D_{t+1} + D_{t+2} )
\]

\[
D_t = E_d (\langle x_{at}, y_{at} \rangle - \langle x_{bt}, y_{bt} \rangle)
\]

\[
D_{t+1} = E_d (\langle x_{at+1}, y_{at+1} \rangle - \langle x_{bt+1}, y_{bt+1} \rangle)
\]

\[
D_{t+2} = E_d (\langle x_{at+2}, y_{at+2} \rangle - \langle x_{bt+2}, y_{bt+2} \rangle)
\]

\{E_d refers to the Euclidean Distance\}

Here \((x_{at}, y_{at})\) and \((x_{bt}, y_{bt})\) represent the spatial attributes or location of the cells \(a\) and \(b\) at the grid or the time-stamp \(t\). Here \(f_2\) is the average Euclidean Distance between \(a\) and \(b\) through grids \(t, t+1\) and \(t+2\).

\[
f_3 = AVG. ( R_t + R_{t+1} )
\]

\[
R_t = 1 : R_{at} \neq R_{bt}
\]

\[\text{else}\]

\[
R_t = 0 : R_{at} = R_{bt}
\]

\[
R_{t+1} = 1 : R_{at+1} \neq R_{bt+1}
\]

\[\text{else}\]

\[
R_{t+1} = 0 : R_{at+1} = R_{bt+1}
\]

\(R_{at}\) and \(R_{bt}\) are the directions the trajectories take from traversing from grid or
time-stamp \( t \) to \( t+1 \). Thus, \( f_3 \) is the average number of direction miss matches in trajectories \( a \) and \( b \). Since any trajectory can have only 2 direction in three grids \( t, t+1 \) and \( t+2 \), from \( t \) to \( t+1 \) and from \( t+1 \) to \( t+2 \) we only have \( R_t \) and \( R_{t+1} \).

We define our global distance function as:

\[
F(S_a, S_b) = \alpha f_1 + \beta f_2 + \gamma f_3
\]

**Distance**

Here we will briefly discuss how to calculate between two “objects. All the distances are based on single linkages.

We have three cases:

1. Distance between a string and a string.

   A regular distance \( F(S_a, S_b) \) between two strings \( S_a \) and \( S_b \)

   ![Fig 5.1: Distance between two strings.](image)

2. Distance between a string and a cluster.

   \[
   F(S_a, C_r) = \text{Min}(F(S_a, S_i)) \quad \text{where} \quad S_i \in C_r
   \]
3. Distance between a cluster and a cluster

\[ F(C_q, C_r) = \text{Min}(F(S_i, S_j)) \text{ where } S_i \in C_q \text{ and } S_j \in C_r \]

Fig 5.2: Distance between a string and a cluster

Fig 5.3: Distance between two clusters

For any string \( a \) to belong to a particular cluster \( C_r \) the distance \( F(S_a, S_{C_r}) \) has to be less than or equal to a user specified threshold value \( \theta \).

Thus we have:

Thus we have:
“For any string \( a \) to belong to a particular cluster \( C \),

\( a \) must satisfy the following condition :

\[
F(S_a, S_C) \leq \theta
\]

Now we will consider the impact of \( \alpha, \beta \) and \( \gamma \) on the kind of patterns we obtain.

The importance of each of these parameters may change from one case to the next. To this end we constrain these three parameters with a condition.

\( \alpha, \beta \) and \( \gamma \) must follow the following constraint :

\[
\alpha + \beta + \gamma = 1
\]

**Importance of \( \alpha \)**

The parameter \( \alpha \) is associated with the temporal attribute of a cell and thus determines the range in which the value \( v \) of the trajectory must fall in order for that particular trajectory to be a part of the cluster under consideration. The higher the value of \( \alpha \) the smaller the range and the smaller the value of \( \alpha \) the larger the range. In deciding the value of \( \alpha \) the user must also keep in mind the quantization did to the data. The way the quantization was done also determines how efficiently the user can manipulate \( \alpha \) to achieve the best possible results. If the number of quanta is too high then the resulting hypotheses may be too specific and not very useful in the real case scenarios and if the range of each quanta is too wide then the resulting hypotheses might be too general for
any use. Thus the effectiveness of the $\alpha$ depends on the quantization of the data and the kind of hypotheses needed.

**Importance of $\beta$**

The parameter $\beta$ is associated with the spatial attribute of the cell. It determines how far a particular trajectory can be from the particular cluster to be included in it. Again, the higher the value of $\beta$ the closer a trajectory has to be to the cluster and lower the value of $\beta$ the further away it may be. The hypotheses obtained with a high value of $\beta$ would provide clusters that are dense and tightly bound. If the dataset is sparse then a high value of $\beta$ would probably give very favorable results.

**Importance of $\gamma$**

$\gamma$ is the simplest and the most straightforward parameter for a user to handle. It is associated with the direction a trajectory takes from one grid to the successive one. In most of the cases the user would want a zero miss match between the trajectory and the particular cluster because any other case the number of resulting hypotheses might increase exponentially in most of the regular datasets. Under normal circumstances the user might want to give it around average value unless he is specifically looking for miss matches. It is important to note that between three grids or time-stamps $f_3$ may take only one three values, i.e., 0, 0.5 or 1. This is because there can be only 0, 1 or 2 miss matches. Thus, even one miss match gives $f_3$ at least a value of 0.5 which in itself is high.
Chapter 6

Approach to the problem

In order to account for the slight discrepancies or errors in the data-set we quantize the data within an interval to a particular value. For example: ‘a’ would represent values from 0-4, ‘b’ would represent values from 5-10, and so on. Moreover for higher data values the interval range is larger than the smaller data values. This is done because the error margin in the higher data values is higher.

Next we assign numbers to each of the 9 directions (Figure 6.1) in which the phenomenon can move.

Fig 6.1: Nine possible directions

Now for the set of 3 layers, we create a string for each of the possible move. For example, string ‘d4b6e’ would represent ‘d’ as the quantized value in the first grid, 4 direction of movement form the first grid to the second, ‘b’ the quantized value in the
second grid, $6$ the direction of movement from the second grid to the third and ‘$e$’ the quantized value in the third grid. We use these strings to extract the patterns from the data-set. The final patterns would contain such strings but of varying lengths.

**Our Methodology**

If we consider that each grid is divided into rectangular cells, then for a particular phenomenon, there are 9 possible ways it can move from one grid to another (Figure 6.1). And from the second layer, for each of the 9 directions there are again 9 directions for the phenomenon to move. Thus if we are to analyze each and every move for just one cell in the first grid, we will have $9 \times 9$ possible moves only till the $3^{rd}$ layer. For $n$ layers, we would have approximately $9^{(n-1)}$ possible moves just for one cell in the first grid. That is why the complexity just shoots up for the increase in the dimensionality of the data-set.

We are currently working on an algorithm which would reduce the complexity considerably by breaking up the data-set and working with 3 grids at a time and extracting the patterns based on the distribution of the data on each grid. After obtaining such patterns from all the sets of 3 continuous grids we can combine the patterns and obtain the final patterns. With such a dynamic programming technique we would not have to deal with all the grids at the same time and we would not have to consider each and every cell on the grid. This method would greatly reduce the complexity of the task.

We start our process by considering only three consecutive grids at a time, for all
possible such sets of grids. For each set of three grids (three time points) we consider all sequences $S_i = <x_i, y_i, t_i, q_i>, <d_2, q_2>, <d_3, q_3>$ and identify all those clusters of sequences $C_i$ which meet the clustering criterion. For example we may say that a $C_i$ is a cluster if:

1. All sequences in it are 3-sequences
2. All of them have same values for $t_i, q_i, <d_{2}, q_{2}>, ..., <d_{3}, q_{3}>$
3. Each cluster has at least some minimum number ($p$) of sequences

The algorithm for finding such clusters is as follows:

```plaintext
procedure subsequencesets(t)
1: CS^3_t ← Ø
2: generatesubsequence(t)
3: for i = 1 to sizeof(St) do
4:    check(i) = 0
5: end for
6: for i = 1 to sizeof(St) do
7:    tempCS^3_t ← Ø
8:    if check(i) = 1 then
9:        continue
10:    end if
11:    check(i) = 1
12:    count ← 0
13:    for j = i to sizeof(St) do
14:        if check(j) = 1 then
15:            continue
16:        end if
17:        check(i) ← 1
18:        if q_{i1:i3}=q_{j1:j3} and d_{i2:i3}=d_{j2:j3} then
19:            count ← count + 1
20:            tempCS^3_t ← tempCS^3_t U St(j)
21:        end if
22:    if count ≥ p then
23:        CS^3_t ← CS^3_t U tempCS^3_t
24:    end if
25: end for
26: end for
```
The Figure 6.2 below shows the 9 directions a trajectory may take from one grid to the next. The Figure 6.3 below shows the exponential increase in the number of trajectories with the increase in the number of grids.

Fig 6.2: 9 possible directions from a cell
As discussed earlier, there is an exponential increase in the number of possible trajectories with the increase in the number of cells. Between two grids there can be 9 possible number of trajectories from all the cells that are not on the edge of the grid as shown in the Figure 6.3. This gives 9 possible cells for the trajectory to traverse through to the next grid. Again from the next grid each cell again gives 9 different possibilities. Thus, between three grids we have 9 * 9 different possibilities for trajectories. Similarly if we have \( t \) grids or time stamps, we have approximately \( 9^{(t-1)} \) possible trajectories.

Let \( CS^3_i \) be the set of clusters \(<C_1, C_2 ... C_n>\) found from the 3 grids beginning at time instant 1. In each of the \( i^{th} \) iteration we consider grids \( i, i+1, i+2 \) and compute \( CS^3_i \). That is, we compute \( CS^3_1, CS^3_2 ... CS^3_{t-2} \). We use form 1 of the string description for the
above algorithm because these are short and contain all the information.

Now, using $CS_3^1$ and $CS_3^2$, we can compute $CS_4^1$ using form 2 of the string descriptions as follows:

1. Combine all those sequences from $CS_3^1$ and $CS_3^2$, which overlap over the grids 2 and 3 and include them in $CS_4^1$. That is, where $x_{i2:i3} = x_{j2:j3}$ and $y_{i2:i3} = y_{j2:j3}$ for $i \in S_3^1$ and $j \in S_3^2$.

2. Each $CS_4^1$ has at least some minimum number ($p$) of length 4 sequences

Algorithm to obtain $CS_4^1$ is as follows:

```plaintext
procedure combine_sequences_sets(t)
1: $CS_{t-1}^t \leftarrow \emptyset$
2: for $i = 1$ to sizeof($CS_{t-1}^t$) do
3:   $temp_1CS_{t-1}^t \leftarrow \emptyset$
4:   for $j = 1$ to sizeof($CS_{t-2}^t$) do
5:     if $x_{(it-2 : it-1)} = x_{(jt-2 : jt-1)}$
       and $y_{(it-2 : it-1)} = y_{(jt-2 : jt-1)}$
then
6:     $temp_1CS_{t-1}^t \leftarrow temp_1CS_{t-1}^t \cup (CS_{t-1}^{t-1}(i) \cup CS_{t-2}^{t-2}(j))$
7:   end if
8: end for
9: for $j = 1$ to sizeof($temp_1CS_{t-1}^t$) do
10:   check($j$) = 0
11: end for
12: for $j = 1$ to sizeof($temp_1CS_{t-1}^t$) do
13:   $temp_2CS_{t-1}^t \leftarrow \emptyset$
14:   if check($j$) = 1 then
15:     continue
16: end if
17:   check($j$) = 1
18:   count = 0
19:   for $k = j$ to sizeof($temp_1CS_{t-1}^t$) do
20:     if check($k$) = 1 then
21:       continue
22: end if
```

40
23: if temp1CS\(_t\)\(_1\)(j) = temp1CS\(_t\)\(_1\)(k) then
24:     temp2CS\(_t\)\(_1\) ← temp2CS\(_t\)\(_1\) U temp1CS\(_t\)\(_1\)(k)
25:     count ← count + 1
26:     check(k) ← 1
27: end if
28: end for
29: if count ≥ p then
30:     CS\(_t\)\(_1\) ← CS\(_t\)\(_1\) U temp2CS\(_t\)
31: end if
32: end for

Similarly, each CS\(_t\)\(_i\) can be computed using CS\(_t\)\(_i-1\) and CS\(_t\)\(_i-2\) as follows:

1. Combine all those sequences from CS\(_t\)\(_i-1\) and CS\(_t\)\(_i-2\) which overlap over the grids \(t-2\) and \(t-1\) and include them in CS\(_t\)\(_i\). That is, where \(x_{it-2:it-1} = x_{jt-2:jt-1}\) and \(y_{it-2:it-1} = y_{jt-2:jt-1}\) for \(i \in CS\(_t\)\(_i-1\)\) and \(j \in CS\(_t\)\(_i-2\)\).

2. Each CS\(_t\)\(_i\) has at least some minimum number (p) of sequences.

Figure 6.4 below shows the process of combining combining two 3-length trajectories to obtain a 4-length trajectory.
Fig 6.4: 3-Length trajectories to 4-length trajectory

The Figure 6.5 and Figure 6.6 below show the process of finding the complete length of the trajectories.

**Combining two 3-clusters to get one 4-cluster**

Fig 6.5: Top-Down Approach to find the length of the pattern.

Fig 6.5: Top-Down Approach
Bottom-Up approach to reflect the actual length to all the substrings.

Fig 6.6: Bottom-Up Approach
Chapter 7

Algorithm

1. The data is quantized into intervals (the intervals depend on the data itself). The quantization is done in such a way that not too many cells have the same quantized value. The number of intervals is denoted by $C$.

2. The data is stored in a 3-D matrix with co-ordinates as $i$, $j$ and $k$ where $i$ is the number of rows, $j$ is the number of columns and $k$ is the number of readings taken (time stamps). The co-ordinates $i$ and $j$ forms a grid and each cell of the grid contains a data item. The time stamps are considered as the number of layers (or the number of grids).

3. Taking three layers at a time we do Sequential Clustering to find the sets of similar substrings that occur more than $n$ times ($n$ is a constant and can be changed for obtaining different results) in that particular set of layers. The distance function used for clustering is $\alpha e + \beta v + \gamma d \leq \Theta$, where $\alpha$, $\beta$, $\gamma$ and $\Theta$ are user defined values and $\alpha + \beta + \gamma = 1$. $e$ is the Euclidean distance between the string and the cluster center. $v$ is the difference between the average value of the string and the cluster mean. $d$ is the average number of direction changes in the string and the cluster.

4. Step 3 is performed for all the $l$-2 sets of layers.

5. The length of all the substrings obtained from steps 3 and 4 is initialized to 3.

6. Going bottom-up, the length of all the overlapping substrings from two sets of consecutive 3-layers is incremented by 1.
Now going top-down the length of the substrings is reflected down to the substrings belonging to the same string or trajectory.

The final strings obtained are plotted in 3 dimensions (time being the 3rd dimension).

The plots can be refined by plotting only those strings that have lengths longer than a specified threshold $l$. 
Chapter 8

Proof

Now, with these basic definitions and functions we will prove that if a pattern exists then our algorithm will find it and also all the patterns extracted by our algorithm actually exist. That is,

“A Global pattern of length k exists if and only if there exist k-2 sub-patterns of length 3 each, such that, they can be combined to form the original Global Pattern”

Case 1:

A Global pattern of length k exists if there exists k-2 sub-patterns of length 3 each (and which satisfy the distance function), such that, they can be combined to form the original Global Pattern.

To prove this case lets assume there are k-2 patterns $S_a^1, S_a^2, ..., S_a^{n-2}$ such that each string satisfies our distance function and when combined the string $S_a = S_a^1 U S_a^2 ... U S_a^{n-2}$ of length k does not satisfy the distance function. Now, $S_a$ can be represented as

$$S_a = T^d U S_a^{n-2}$$

Where $T^d = S_a^1 U S_a^2 ... U S_a^{n-3}$. Now since $S_a^{n-2}$ satisfies the distance function and $S_a$ does not, then $T^d$ cannot satisfy the distance function. Further $T^d$ can be represented as

$$T^d = T^2 U S_a^{n-3}$$
Where $T^2 = S^1_a \ U \ S^2_a \ ... \ U \ S^{n-4}_a$. With the similar logic $T^2$ cannot satisfy the distance function. Similar if we keep dividing the global sting then we will find that $T^{n-2} = S^1_a$ cannot satisfy the distance function which is a contradiction.

**Case 2:**

*A Global pattern of length k exists only if there exists k-2 sub-patterns of length 3 each (and which satisfy the distance function), such that, they can be combined to form the original Global Pattern.*

To prove this lets assume there is a pattern of length k, $S_a = S^1_a \ U \ S^2_a \ ... \ U \ S^{n-2}_a$ which satisfies our distance function but at least one of the sub-strings from $S^1_a, S^2_a \ ... \ S^{n-2}_a$ does not. Lets represent $S_a$ as $S_a = T^l \ U \ S^{n-2}_a$ where $T^l = S^1_a \ U \ S^2_a \ ... \ U \ S^{n-3}_a$. This shows that both $T^l$ and $S^{n-2}_a$ have to satisfy the distance function. Now using the same method as in case 1 we can show that each of the sub-string has to satisfy the distance function for the global pattern to satisfy the distance function.

"if there exists a global pattern then our algorithm can extract it"

Let’s assume that there is a pattern of length $k$ which exists globally and which cannot be extracted by our algorithm. Let the pattern consist of two strings be $S_a$ and $S_b$ which are represented as:
Now, since $S_a$ and $S_b$ belong to the same cluster, our two distance functions must be satisfied. That is,

$$f_1 = |q_{at} - q_{bt}| + |q_{at+1} - q_{bt+1}| + \ldots + |q_{at+n-1} - q_{bt+n-1}| \leq \lambda$$

$$f_2 = E_d(<x_{at}, y_{at}>, <x_{bt}, y_{bt}> ) \leq \lambda$$

Now we can divide the strings $S_a$ and $S_b$ such that each one of them is a union of $n-2$ sub-strings of length 3 each. That is,

$$S_a = S_a^1 \cup S_a^2 \ldots \cup S_a^{n-2}$$

$$S_b = S_b^1 \cup S_b^2 \ldots \cup S_b^{n-2}$$

Where, $S_i^j$ is represented as

$$S_a^j \rightarrow <x_{ai}, y_{ai}, q_{ai}>, <d_{ai+1}, q_{ai+1}>, <d_{ai+2}, q_{ai+2}>, \ldots $$

Now, since $S_a$ and $S_b$ belong to the same cluster and satisfy the three distance functions, the each of the $S_a^j$ and $S_b^j$ must also adhere to the same conditions. That is,

$$|q_{at} - q_{bt}| + |q_{at+1} - q_{bt+1}| + |q_{at+2} - q_{bt+2}| \leq \lambda \quad \rightarrow \quad 1a$$

$$|q_{at+3} - q_{bt+3}| + |q_{at+4} - q_{bt+4}| + |q_{at+5} - q_{bt+5}| \leq \lambda \quad \ldots \quad \rightarrow \quad 1b$$

$$|q_{at+n-3} - q_{bt+n-3}| + |q_{at+n-2} - q_{bt+n-2}| + |q_{at+n-1} - q_{bt+n-1}| \leq \lambda \quad \rightarrow \quad 1c$$

And

$$E_d(<x_{at}, y_{at}>, <x_{bt}, y_{bt}>) \leq \lambda, \quad \rightarrow \quad 2a$$

$$E_d(<x_{at+3}, y_{at+3}>, <x_{bt+3}, y_{bt+3}>) \leq \lambda \quad \ldots \quad \rightarrow \quad 2b$$

$$E_d(<x_{at+n-3}, y_{at+n-3}>, <x_{bt+n-3}, y_{bt+n-3}>) \leq \lambda \quad \rightarrow \quad 2c$$
Now, if we combine 1a 2a 3a and 1b 2b 3b, we can see that they themselves satisfy our distance functions. This implies that each set \( \{S_a^1, S_b^1\}, \{S_a^2, S_b^2\}, \ldots, \{S_a^{k-2}, S_b^{k-2}\} \) form a “sub-cluster” of their own. Now, since these “sub-clusters” satisfy the distance functions then they have to be detected by our algorithm. Thus, we can say that if there exists a global pattern then our algorithm can extract it.
Chapter 9

Complexity Analysis

Theoretical Complexity

Complexity without our algorithm

No. of grids = N

Grid size = X * Y

No. of strings generated ≈ 9^{(N-1)} * (X * Y) → Let this number be $S_n$

Length of each string = N + (the number of directions in each string) = N + (N – 1) = 2N – 1

No. of strings comparison = ($S_n$ -1) + ($S_n$ – 2) + … + 1 = $[S_n * (S_n - 1)]/2$

Total no. of comparisons needed

= (2N – 1) * $[S_n * (S_n - 1)]/2$

= (2N – 1) * $[(9^{(N-1)} * (X * Y)) * (9^{(N-1)} * (X * Y) - 1) ]/ 2$

The order of complexity = $X^2 * Y^2 * N * 9^{2(N-1)}$

Complexity of our Algorithm

No. of grids = N

Grid size = X * Y

No. of strings generated in one set after dividing the grids in sets of 3 = $9^2 * X * Y$

→ Let this number be $S_m$
Length of each string = 5

Coordinates needed for comparison = 3

No. of strings comparisons = $S_m^2 = 3 \times 9^4 \times (X \times Y)^2$

Total no. of comparisons needed = $S_m^2 \times (N-2) = 3 \times 9^4 \times (N-2) \times (X \times Y)^2$

The order of complexity = $N \times X^2 \times Y^2$

**Actual Complexity**

**Pattern Type:** 1

**Module:** 1

Number of grids = N

Size of each grid = $X \times Y$

No. of comparisons for direction for each string = 28

No. of initial assignments = 12

Total number of initial operations for each set of grid = $X \times Y \times 9 \times (28 + 12)$

\[= 360 \times X \times Y\]

Evaluation operations = $(0.5 \times X \times Y \times (1 + X \times Y) \times 4) + X \times Y$

\[= 3 \times X \times Y + (X \times Y)^2\]

Total No. Of operations = $(360 \times X \times Y) + [3 \times X \times Y + (X \times Y)^2]$

\[= 362 \times X \times Y + (X \times Y)^2 \rightarrow S_n\]

Total No. of operations including all the sets of grids = $(N-2) \times S_n$

\[= 362 \times X \times Y \times (N-2) + (N-2) \times (X \times Y)^2\]
Module: 2

Let the maximum number of strings selected in any set of 3 Grids be $c$

Total number of operations = $2 \times 5 \times (N - 2) \times c^2$

Thus, the total complexity = $(N - 2) \times [362 \times X \times Y + (X \times Y)^2 + 10 \times c^2]$

Order of complexity = $N \times (X \times Y)^2$
Chapter 10

Related Research

In Data Mining, for many years the researches have felt the need for simultaneous treatment of spatial and temporal aspects [1]. Spatial and temporal databases have been studied separately in detail for a long time, but it has been only recently that the spatio-temporal data has come under observation. [2] contains a comprehensive bibliography on spatio-temporal databases till 1994. Even though spatio-temporal databases have gained some attention in the last few years, there still remains a lot to be achieved in the area.


Research done on spatio-temporal data mining mainly focuses on two major areas. Much work has been done on mining and clustering of mobile objects in the spatio-temporal context. These objects make trajectories in time in a 2-dimensional plane. The goal of such research is to cluster those objects or trajectories which follow the same or similar path according some metric of similarity. Vlachos et. al. in (M. Vlachos 2002), use the LCSS (Longest Common Subsequence) measure of similarity for clustering similar trajectories. They use projections of subsequences of two trajectories to find the similarity between them. Buzan et. al. in (Buzen & Sclaroff 2004) also use the LCSS metric to cluster motion trajectories in video. Their method can cluster similar trajectories of varying lengths by reducing the length of the longer trajectories and matching only
those parts which are similar. Mamoulis et. al. in (N. Mamoulis & Cheung 2004), use the \textit{Apriori} algorithm (Agrawal & Srikant 1994) for clustering trajectories obtained from historical data. They consider corresponding points of various trajectories and cluster them depending on whether they all belong to a small spatial neighborhood or not. A paper published by Ilias Tsoukatos and Dimitrios Gunopolis presents a Depth-First-Search-like approach for fast mining of frequent spatiotemporal patterns in environmental data. In all the above cases the database comes in the form of known or observed trajectories. Not much work has been done in the area of finding patterns in a dataset of natural phenomena like weather etc. which has not been reduced to a subset of interesting trajectories but consists of an implicit set of a very large number of possible trajectories. Some work has been done along these lines but that focuses on some very specific situational cases (Stern 2004; McGregor 1996). In his work (Kitamoto) on spatio-temporal data, he specifically focuses on discovering useful knowledge from the large collection of satellite images of Typhoons. Jeremy Mennis and Jun Wei Liu did a case study which used association rule mining to explore the spatial and temporal relationships among a set of variables that characterize socioeconomic and land cover change in the Denver, Colorado, USA region from 1970–1990. In their paper (Kalnis, Mamoulis and Bakiras), they study the discovery of moving clusters in a database of object trajectories. In their problem the identity of
each cluster remains the same but their location and content may change over time. Peuquet and Duan (Peuquet and Duan 1995), recorded the evolution of thematic data for a given geographic area over time using a time based method that involved event chains. The algorithm we present works on a whole area rather than a few specific points in the area. No data about the interesting trajectories is required.

In 2000, Leakha M. Henry and Brett A. Bryan (Visualizing the Spatio-Temporal patterns of Motor Vehicle Theft in Adelaide, South Australia), studied the behavior of Motor Vehicle Theft (MVT) using various Spatial Techniques to identify the vehicle theft “hotspots”. Their studies revealed that the “distribution of MVT in Adelaide for 1999 is not uniform but displays distinct patterns in space and over time”. The mining of such Spatio-Temporal patterns would definitely help in the prediction of behavior of MVT for the next few months.
Chapter 11

Results

Here we present various results illustrating the different clusters which can be discovered by our algorithm.

Due to the absence of any specific algorithm capable of discovering such clusters we compare the results with a simple algorithm which would use exhaustive search to discover similar clusters. We use the size of the search space as the basis of comparison. $Q(n_1)$ is the approximate number of strings in the search space of our algorithm and $Q(n_2)$ is the approximate number of strings in the search space of the simple algorithm used for comparison.

To illustrate we consider a general case here. Let there be $t$ number of grids or time-stamps. Let the size of each grid be $x^y$. Then,

$Q(n_1) = (t-2) \times (x^y) \times 9^2$

$Q(n_2) = (x^y) \times 9^{(t-1)}$

And

$Q(n_1) : Q(n_2) = (t-2) : 9^{(t-3)}$
Various Patterns

Firstly we will discuss the results obtained from various datasets containing the different kind of patterns that we discussed earlier. The sizes of various datasets are shown as $(X, Y, G)$, that is, $X$ and $Y$ are the x-y dimensions of each grid and $G$ is the number of grids.

Data-Set 1:

Size: $(10, 10, 10)$

Description: Close and Distant patterns.
Fig 11.1: Output-Close and Distant Patterns

\[ \frac{Q(n_1)}{Q(n_2)} = 1 : 5.98 \times 10^4 \]

**Data-Set 2:**

**Size:** (10, 10, 10)

**Description:** Clusters of varying sizes.
Fig 11.2: Output-Patterns varying sizes

\[ Q(n_1) : Q(n_2) = 2 : 9^{16} \]

**Data-Set 3:**

**Size:** \((10, 10, 10)\)

**Description:** Various overlapping clusters containing various patterns of different lengths and properties.
Fig 11.3: Output-Overlapping Patterns

\[ Q(n_1) : Q(n_2) = 1 : 3.87 \times 9^5 \]

Data-Set 4:

Size: \((10, 10, 10)\)

Description: Cluster containing one long pattern which includes small trajectories branching off from the main pattern.
Fig 11.4: Output-Pattern with branches

\[ Q(n_1) : Q(n_2) = 1 : 3.87 \times 9^5 \]

**Data-Set 5:**

**Size**: \((10, 10, 10)\)

**Description**: Two intertwined patterns.
Fig 11.5: Output-Intertwined Patterns

\[ Q(n_1) : Q(n_2) = 1 : 3.87*9^7 \]

**Data-Set 6:**

**Size:** (100, 100, 50)

**Description:** Three periodic patterns with different periods.
Fig 11.6: Output-Periodic Patterns

\[ Q(n_1) : Q(n_2) = 2 : 9^{16} \]

**Data-Set 7:**

**Size:** (10, 10, 10)

**Description:** Dense and Sparse patterns.
Fig 11.7: Output-Dense and Sparse Patterns

\[ Q(n_1) : Q(n_2) = 2 : 9^{16} \]
**Data-Set 8:**

**Size:** (25, 25, 4)

**Description:** Dataset with noise and missing data.

**Input:** Our Input data consists of four grids each of which size of 25X25. Each grid contains a few clusters scattered on it. These clusters change their position, shape and size as we progress from one grid to the next. The four grids and their clusters and the progression of these clusters across these grids are shown below. The color in each cell represents its temporal attribute at that particular time stamp. Any two cells with same color indicate that those two cells have the same temporal attribute.

![Input-Grid 1](image1.png)

**Fig 11.8:** Input-Grid 1
Fig 11.9 Input-Grid 2

Fig 11.10: Input-Grid 3
As we can see from these plots, there are three prominent clusters moving across these grids.

Output: \[ Q(n_1) : Q(n_2) = 1 : 4.5 \]
Fig 11.12: Output- $\alpha=0.4$, $\beta=0.4$, $\gamma=0.2$, $\theta=0.08$, $Length \geq 3$.

Fig 11.13: Output- $\alpha = 0.4$; $\beta = 0.4$; $\gamma = 0.2$; $\theta=0.08$; $length = 4$
Fig 11.14: Output- $\alpha = 0.2; \beta = 0.3; \gamma = 0.5; \theta = 0.08; \text{length} \geq 3$

Fig 11.15: Output- $\alpha = 0.2; \beta = 0.3; \gamma = 0.5; \theta = 0.08; \text{length} = 4$
Fig 11.16: Output- $\alpha = 0.3; \beta = 0.6; \gamma = 0.1; \theta = 0.08; length \geq 3$

Fig 11.17: Output- $\alpha = 0.3; \beta = 0.6; \gamma = 0.1; \theta = 0.08; length = 4$
References


