Pak. J. Biotechnol. Vol. 13 (special issue on Innovations in information Embedded and Communication Systems) Pp. 178 - 183 (2016)

A FRAMEWORK FOR MINING UNIFYING TRAJECTORY PATTERNS USING SPATIO-TEMPORAL DATASETS BASED ON VARYING TEMPORAL TIGHTNESS

Rahila R.[1] and R, Siva[2]

Department of CSE, KCG College of Technology, Chennai, India
[1]rahila398@gmail.com; [2]sivar@kcgcollege.com

ABSTRACT

Trajectory patterns discovery is useful in learning interactions among moving objects. Different types of trajectory patterns such as flock patterns, convoy patterns and swarm patterns have been proposed earlier, but methods were developed for mining only a particular type of trajectory patterns. The pattern discovery becomes difficult and inefficient as users typically may not know which types of trajectory patterns are present hidden in their data sets. One main observation is that trajectory patterns can be arranged based on the strength of temporal tightness. In this paper, a framework of mining unifying trajectory patterns also known as UT-patterns based on varying temporal tightness is proposed. The framework consists of three phases: initial pattern identification, granularity adjustment, classification and visualization. The preprocessing is done by using trajectory clustering algorithm and a set of initial UT-patterns identification are done in the first phase by using the spatio-temporal datasets. The granularity adjustments i.e., levels of detail are adjusted by drill down and roll up to detect other types of UT-patterns in the second phase. Classifications of the UT-patterns are done using the Trajectory classification algorithm in the third phase. Visualization of the classified UT-patterns is done in the final phase according to the patterns obtained as the result of classification.

Keywords— Trajectory patterns, temporal tightness, UT-patterns.

I. INTRODUCTION

Rapid enhancement in the modern positioning technology enables large scale collection of various moving objects data such as animal movement data, vehicle movement data, and mobile tracking data. The data are usually collected from satellite, sensor, and other wireless technologies. For instance, animal scientists attach sensor tags on animal to track their movement and mobile phones have enabled the tracking of almost all kind of data.

The groups of object that move show both synchronous and asynchronous movement patterns. Synchronous movement patterns are shown in moving objects that often interact with each other: for example, a set of objects moving together or set of moving objects that chase another set of moving objects with some time delay. On the other hand, asynchronous movement patterns: for example,a set of objects moving may follow the same path each year. Such trajectory patterns are collectively called as unifying trajectory patterns (UT-patterns) i.e. unifying trajectory pattern that shows different temporal tightness. Traje-ctory patterns are the path or trace of moving object patterns. In general Trajectories are sequences that contain the spatial and temporal information about move-ments. Trajectories are usually given as spatiotemporal (ST) sequences: $<(x_0, y_0, t_0)\ldots (x_n, y_n, t_n)>$. Where x_i, y_i is the position coordinate relative to the origin and t_i is the time stamp for the position information.

UT-patterns are defined informally as set of moving objects that are nearly related in terms of location, time or both. An example of UT-patterns can be explained in deer migration as a herd of deer that move together at the same time and thus they are close to each other. Another example is wolf predatory behavior on wild hoofed mammals where the wolves chase prey over long distance in open space.

Majority of applications can benefit from the UT-patterns mining. For example, in animal trajectories, zoologists can learn migration or movement patterns of animals such as deer and wolf. In battleship trajectories, commanders can discover movement of the enemy. In soccer-player trajectories, attack strategies of the opponent team can be guessed by the coaches and so on.

Notable efforts have been devoted to trajectory patterns discovery in data mining and computational geometry. A list of well-known studies includes flock patterns [11], convoy patterns [2], swarm patterns [18] [19], and sub-trajectory clusters [7]. Observed that each of them corresponds to just one type in a broad range of trajectory patterns, and earlier methods have been developed only to handle one specific type. In some of the trajectory patterns both spatial and temporal constraints are considered. But these are not considered by the existing techniques. This limitation make discovery of pattern inefficient since users may not know which types of trajectory patterns are present or hidden in their data sets. For example, a data set may contain sets of moving objects that arrived at various locations within various time intervals. These are mined to identify the interesting patterns.

One best way of classifying existing trajectory patterns is to consider the rigidity of temporal restrict-tions on the patterns. The observation motivates the study which develops a framework that have the capa-bility of navigating the patterns at all different levels of temporal constraints.

In summary, the major contributions of this paper are given as follows:

- To develop a framework of mining trajectory patterns based on varying temporal tightness, which classify the UT-patterns into three types depending on the strength of temporal constraints.
- To present an algorithm TRACLUS [7] that is highly efficient for data preprocessing.

- To present an algorithm for discovering a good set of initial UT-patterns where both spatial and temporal constrains are checked.
- To present an algorithm to find remaining UT-patterns using the reference movement during initial pattern identification.
- To improve the efficiency of the unifying trajectory pattern mining using trajectory classification algorithm [9] and visualizing the results based on the classified result using Google map [3] or Google earth [4].

II. RELATED WORK

Phan Nhat Hai et al proposed GeT Move: a parameter free unifying incremental spatiotemporal patternmining approach for extracting different kinds of patterns i.e. [11,2,18] and [7]. Part of this approach is based on an existing state-of-the-art algorithm that is extended to take advantage of well-known frequent closed dataset mining algorithms. In order to use it, first redefine spatio-temporal patterns in a dataset context. Secondly, a spatio-temporal matrix is proposed to describe original data and then an incremental frequent closed dataset-based spatio-temporal patternmining algorithm to extract patterns. The main idea is to rearrange the input data based on nested concept so that incremental GeT Move can automatically extract patterns efficiently without parameters setting.Object movements are quite complex and therefore data can be unclean or noisy. So preprocessing is needed to collect and clean the raw data and to interpolate missing points in [15].

Zhenhui Li et al proposed a moving object data mining system, MoveMine that include multiple data mining functions [6,17]. Two moving object pattern mining functions are newly developed they are: (1) periodic behavior mining and (2) swarm pattern mining. In periodic behaviors mining a reference location-based method is developed, which initially detects the reference locations, discovers the periods present in complex movements, and then finally find periodic patterns by hierarchical clustering. In swarm patterns mining a highly efficient technique is developed that uncover flexible moving object clusters by relaxing the well known enforced collective movement constraints in [19].

Jae-Gil Lee et al [8] proposed a partition and detect framework for trajectory outlier detection, which partitions a trajectory into a set of line segments, then detects outlying line segments for trajectory outliers. The primary advantage of the framework is to detect outlying sub-trajectories from a trajectory database. Based on this partition and detect framework, a trajectory outlier detection algorithm TRAOD is developed. The algorithm consists of two phases: partitioning and detection. For the first phase, a two-level trajectory partitioning strategy is proposed that ensures both high quality and efficiency. For the second phase, a hybrid of the distance-based and density-based approaches is proposed. Experimental results show that TRAOD can correctly detects outlying sub-trajectories from real trajectory data in [8].

Yan Huang et al proposed a framework for mining sequential patterns from spatio-temporal event datasets. The main part of the framework is the use of a sequence index as the significance measure for spatio-temporal sequential patterns. It defines a density ratio for two sequences and then extends it to sequence index for longer sequences. In addition, statistical interpretation is proposed for the proposed sequence index. Then a novel algorithm called Slicing-STS-Miner for sequential pattern mining is applied onto the spatio-temporal datasets. This algorithm handles the challenges by using the sequence index which not guarantees the downward closure property. Slicing-STS-Miner uses the temporal slicing to partition the dataset as overlapping slices according to time when the number of events is very large to be processed in memory. It processes each time slice separately and uses the unidirectional property of time to recover the sequential patterns across slice boundaries with low cost. Algebraic costs of the algorithms is analyzed and the performance of the proposed algorithm experimentally evaluated against a simple algorithm called STS-Miner, which uses the weak monotone property of the sequence index on both synthetic and real-world datasets in [16].

The classification of the trajectory patterns by considering the temporal tightness includes flock [11], convoys [2], or moving clusters [14], at one extreme where objects move together at the same time. A flock [11] in a time interval I, where the duration of I at least k, consists of at least m entities such that for every point in time within I, there exist a disk of radius r that contains all the m entities. The convoy [2] is an extension of the flock using the concept of density-based clustering. A moving cluster [14] is a sequence of (snapshot) clusters c_1, c_2 … c_k such that for each timestamp i $(1 \leq i < k)$, c_i and c_{i+1} share a sufficient number of common objects.

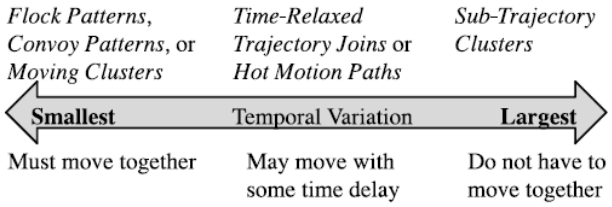

Figure 1 A spectrum of existing trajectory patterns [10].

Time-relaxed trajectory joins [12] or hot motion paths [1] are placed at in between the extremes where objects may follow other objects with some time delay. Two trajectories are time-relaxed joined if there exist time intervals of the same length δ_t such that the distance between the locations of the two trajectories during these intervals is no more than a spatial threshold ε, and the relative matching between the two trajectories occurs within some time distance. A hot motion path is a route that is frequently followed by multiple objects within a tolerance margin ε in the last W timestamps.

Sub-trajectory clusters [7] is a trajectory is partitioned into a set of line segments, and these trajectory partitions are grouped as a cluster according to their spatial similarities only. That is, a sub-trajectory cluster

is a set of trajectory partitions which are very close to each other and are heading to a similar direction. Here, temporal information is not considered. Since UT-patterns are defined differently from the existing patterns, it does not really subsume or incorporate any of the existing patterns. The most prominent advantage of it is that it can fall into any of the three types in Figure 1, whereas existing patterns may fall into only one of the three types. Contrary to the patterns in Figure 1, there are several other patterns that do not force strong spatial constraints. For example, a swarm [18] [19] is a group of moving objects that contains at least min_o individuals who are in the same cluster for at least min_t timestamp snapshots. Since some moving objects are allowed to leave the cluster for some period of time, the intermediate paths between timestamps are not very important, pretty much different from a UT-pattern.

III. PROPOSED FRAMEWORK

The proposed framework consists of five different phases: 1) Initial pattern identification, 2) Granularity adjustments, 3) Classification and Visualization. The input to the framework is the set of spatio-temporal trajectory datasets. These spatio-temporal datasets [5] are preprocessed by using the trajectory clustering algorithm of partition and group framework [7]. As indicated by the name the framework consists of two phases: partitioning and grouping as in Figure 2.

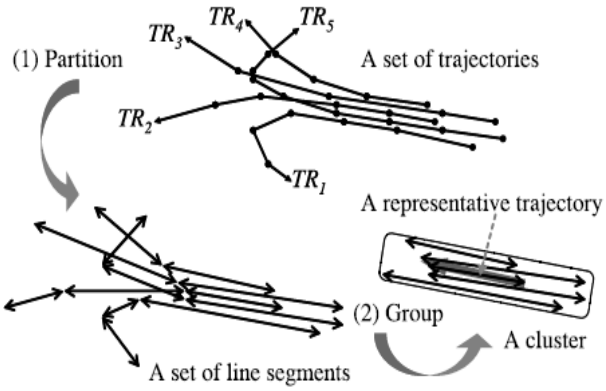

Figure 2 Example of Trajectory Clustering in partition and group framework [7].

In *partitioning* phase each trajectory is optimally partitioned into a set of line segments. These line segments are provided to the next phase. The *grouping* phase group similar line segments into a cluster. Here, a density-based clustering method is exploited. The advantage of the partition-and-group framework is the discovery of common *sub*-trajectories from a trajectory database. This is the exact reason for partitioning a trajectory into a set of line segments.

The input of the framework is a set of trajectories $I = \{TR1,\ldots,TR_{num_{tra}}\}$. A trajectory is a sequence of the locations of an object at each timestamp and is denoted as $TR_i = p_1p_2 \ldots p_j \ldots p_{len_i}$. Here, p_j $(1 \le j \le len_i)$ is a point $(x_j; y_j; t_j)$ in a three dimensional space, where $(x_j; y_j)$ indicates the location of the object at the time t_j. The length len_i of a trajectory can be identically different from those of other trajectories. A trajectory $p_{c_1}p_{c_2} \ldots p_{c_k}$ $(1 \le c1 < c2 < \ldots < ck \le len_i$, where $c_k = c_{k-1} + 1)$ is called a sub-trajectory of TR_i. A trajectory partition is a line segment p_ip_j $(i < j)$, where p_i and p_j be the points chosen from the same trajectory. The output of first two phases is a set of UT-patterns $O = \{UT_1,\ldots,UT_{num_{pat}}\}$. A UT-pattern is a set of trajectory partitions with a reference movement. A trajectory partition which belongs to the same UT-pattern is close in terms of location and time according to a similarity function. A reference movement is the line segment, where the first point is the starting location and time of the overall movement, the second point is the ending location and time.

In summary, a UT-pattern is $UT_i = (R_i, L_i)$, where R_i is the reference movement, and L_i the set of trajectory partitions. This output is given as input to the trajectory classification algorithm that uses the trajectory based clustering technique. The procedure of trajectory-based clustering is similar to the traclass framework proposed by Lee et al., [9]. The partition-and-group framework was extended for classification purposes such that the class labels are incorporated into clustering.

The classified outputs of various trajectory patterns are finally visualized by using the Google earth or Google map. The input to the visualization phase is the UT-patterns which are classified according to its temporal tightness. To overcome the limitation present in the existing classification system the UT-patterns are broadly classified as time restricted, time delayed and time unrestricted patterns in figure 3.

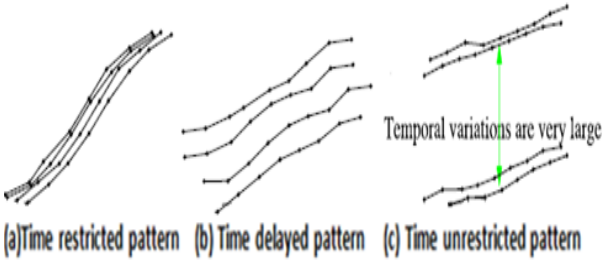

Figure 3 Three Types of pattern classification similar to [10].

Algorithm 1: UT-pattern mining

Input: A dataset of spatio-temporal trajectories $I = \{TR1,\ldots, TR_{num_{tra}}\}$

Output: A set of classified UT-patterns visualized using Google earth or Google map application programming interface.

1: /* Phase 1: Initial pattern identification as in [10]*/
2: /* Perform sub trajectory clustering over I for pre-processing the spatio-temporal datasets [5] using partition and group framework [7].*/
3: /* Partitioning Phase */
4: for each ($TR \in I$) do
5: Execute Approximate Trajectory Partitioning;
6: Get a set L of line segments using the result;
7: Accumulate L into a set D;
8: /* Grouping Phase */
9: Execute Line Segment Clustering for D;
10: Get a set $O=\{C1,..,C_{num_{clus}}\}$ of clusters as the result;
11: Get a set Call of sub-trajectory clusters;
12: for each $C \in C_{all}$ do

13: /* Algorithm 2 */
14: Execute Initial Pattern Identification over C;
15: Get a set P of UT-patterns as the result;
16: Accumulate P into a set P_{all};
17: end for
18: /* Phase 2: Granularity Adjustment */
19: /* Algorithm 3 */
20: Execute Pattern Forest Construction over P_{all} [10];
21: Return the set of UT-patterns in the forest;
22: /*Phase 3: Classification Visualization of UT-patterns*/
23: Classify the UT-patterns into the three types based on its temporal cohesion [10];
24: The UT-patterns that move together are labelled as time restricted patterns;
25: UT-patterns that move together with some time delay are labelled as time delayed patterns;
26: The UT-patterns that doesn't follow any time restrictions and move as it is in the same trajectory partition are labelled as time unrestricted.
27: Visualization of the classified UT-patterns in Google earth [3] or Google map [4];

The phase of initial pattern identification is processed in two steps. First, sub-trajectory clusters in the (X, Y)-space are discovered using the partition-and-group framework [7] without temporal constraints consideration. A sub-trajectory cluster is a set of trajectory partitions that move close together in a same direction. Then, for every sub-trajectory cluster, initial UT-patterns in the (location, Time)-spaces are retrieved by considering both spatial and temporal constraints. The location of the trajectory partitions within a particular sub-trajectory cluster can be represented using single dimension since their directions are very similar. The advantage of this stepwise approach is to improve efficiency for the following reasons. First, the dimensions are reduced from three to two. Second, the search space is reduced to a certain sub-trajectory cluster, not the entire dataset.

Algorithm 2: Initial pattern identification
Input: A set L of trajectory partitions in a cluster C
Output: A set P of initial UT-patterns
1: $\mathbb{L} \Leftarrow L$, $R_1 \Leftarrow$ DeriveRefMovement($\mathbb{L}_1$);
2: $P \Leftarrow \{(R_1, \mathbb{L}_1)\}$;
3: repeat
4: Choose the mth UT-pattern from P, where
$$m = \underset{(R_m, L_m) \in P}{\operatorname{argmax}} C(R_m, L_m)$$
5: /*Split the mth UT-pattern into two splits*/
6: Choose the pair of trajectory partitions, where
$$(L_p, L_q) = \underset{L_p, L_q \in L_m}{\operatorname{argmax}} dist(L_p, L_q);$$
7: /*Distribute t-partitions of $\mathbb{L}_m$ into two*/
8: /*Derive a reference movement for each split*/
9: $R_m^p \Leftarrow$ DeriveRefMovement(L_m^p);
10: $R_m^q \Leftarrow$ DeriveRefMovement(L_m^q);
11: /*Replace the mth pattern by the new ones*/
12: /*Check if L(H) + L(D|H) decreases*/
13: if MDL(P') < MDL(P) then
14: $P \Leftarrow P'$;
15: end if
16: until MDL(P') ≥ MDL(P)
17: Return the set P of initial UT-patterns;
18: function DeriveRefMovement($\mathbb{L}_k$)
19: /* Consider each t-partition as a candidate */
20: /* Find the one that minimizes the code length */
21: Return the sth candidate R_k^s, where
$$s = \underset{R_k^s \in R_k}{\operatorname{argmin}} C(R_k^s, L_k);$$
22: end function

The initial pattern detection or identification algorithm uses the MDL cost function [13]. The MDL cost consists of two components [13]: L(H) and L(D|H). Here, H is the hypothesis and D the data. The two components are informally stated as: L(H) is the length of the description of the hypothesis represented in bits; and L(D|H) is the length of the description of the data when encoded with the use of the hypothesis represented in bits. The best hypothesis H to explain D is the one that minimizes the sum of L(H) and L(D|H). The MDL principle fits very well in this problem. H corresponds to a set of UT-patterns, and D the set of trajectory partitions. As a result, finding a good set of UT-patterns translates to finding the best hypothesis using the MDL principle.

Even though the initial set best captures the underlying patterns of trajectory partitions based on the MDL cost, some users may still need to find smaller (finer) or larger (coarser) patterns. This motivates the phase of granularity adjustment. Split and merger of UT-patterns are semantically analogous to drill down and roll up [10] widely used in OLAP. However, the operations are more complex since the drill-down and roll-up dimensions are not specified by the users but are automatically defined to be either location or time. This choice is clearly formulated using the almost same information-theoretic formula [13] as in the first phase. Moreover, as far as known, there has been no work for automatically building concept hierarchies for spatio-temporal patterns.

Drill down is done only if the split decreases the MDL cost as Algorithm 2. Its main objective is to derive multiple time restricted (or smaller time delayed) patterns from a time delayed pattern. In other words zoom in is performed. The drill down dimensions is selected automatically so as to minimize the MDL cost [13]. Roll-up is the reverse operation of drill-down.

Algorithm 3: Granularity Adjustment
Input: A set P_{all} of initial UT-patterns
Output: A pattern forest FR
1: FR ⇐ P_{all}, Q ⇐ P_{all}; /*Q is a queue*/
2: /*1. Drill Down*/
3: while Q ≠ Ø do
4: Pop a UT-pattern UT_i from Q;
5: if UT_i can be split into UT_i^1 and UT_i^2 then
6: Push UT_i^1 and UT_i^2 into Q;
7: /* Update the pattern forest */

8: Add two vertexes for UT^1_i and UT^2_i into FR;
9: Add two edges for (UT_i, UT^1_i) and (UT_i, UT^2_i) into FR;
10: end if
11: end while
12: /* 2. Roll Up */
13: /* P_c is a set of UT-patterns in the cth cluster */
14: for each $P_c \subseteq P_{all}$ do
15: for each pair of $UT_i \in P_c$ and $UT_j \in P_c$ do
16: if UT_i and UT_j can be merged into UT_{ij} then
17: Add UT_{ij} into P_c;
18: /* Update the pattern forest */
19: Add one vertex for UT_{ij} into FR;
20: Add two edges for (UT_{ij} UT_i) and (UT_{ij},UT_j) into FR;
21: end if
22: end for
23: end for
24: Return the pattern forest FR;

The classification and visualization of the UT-patterns is done in the final phase i.e. the third phase of framework. The visualization can be done by plotting the UT-patterns on the 2D plane or embedding into other visualization tools (e.g., Google map and Google earth).The output written in Google map or Google earth format can help users to explore the framework results. Furthermore, users can also compare the results of varied datasets to understand the differences between object movement behaviors. Moreover, different patterns can be visualized in the same time on the same map.

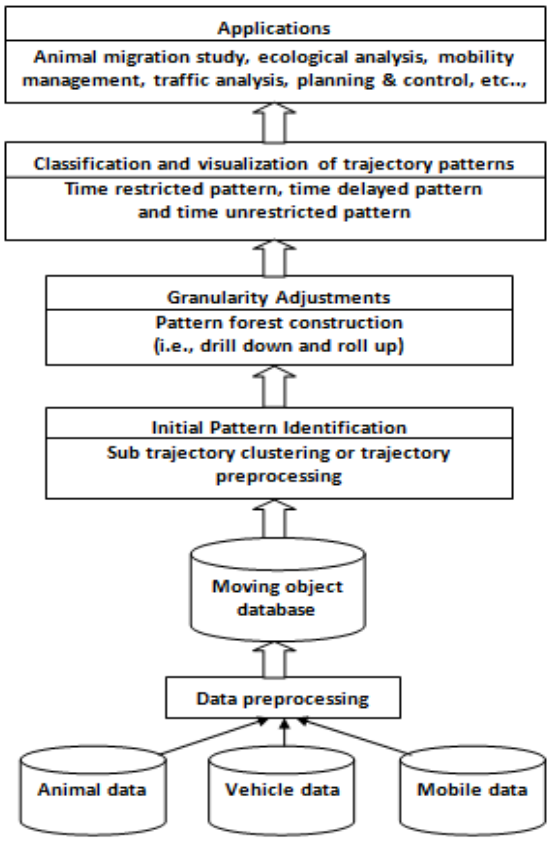

Figure 4 Architecture of UT-pattern mining framework.

The overall architecture of the framework in Figure 4 starts with moving object datasets collection. The datasets may include animal data, vehicle data or mobile data. The datasets are cleaned by preprocessing then they are stored in moving object database which may include both spatial and temporal attributes. Group of these datasets are given as input in the form of trajectories to initial pattern identification. The output obtained in this phase is given as input to granularity adjustment. Then the output of granularity adjustment can be used for classification. Finally the visualized results of UT-patterns can be used for applications such as animal migration study, ecological analysis, mobility management, traffic analysis, planning and control etc.

IV. PERFORMANCE ISSUES AND MEASURES

The analysis of the issues which are present in the performance of the framework is calculated based on the computational time and accuracy to the size of the datasets and UT-patterns.

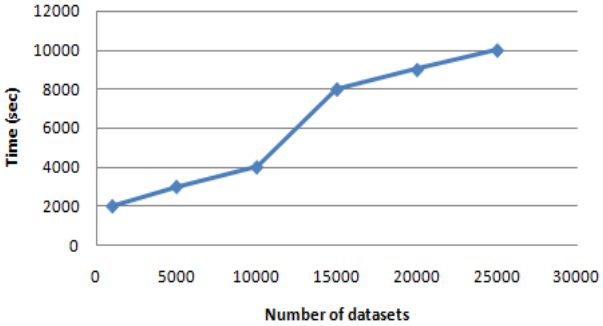

Figure 5 Performance based on number of datasets used to the computational time.

The time of computation increases as the number of datasets $t_{clustering}$ increases linearly, whereas $t_{initial+forest}$ exhibits quadratic growth as the data size increases is shown in Figure 5. This quadratic growth will decreases as the degree of parallelism increases using more CPU cores. Since initial pattern receives only the trajectory partitions present in clusters, processing time is affected by the proportion of trajectory partitions belonging to clusters.

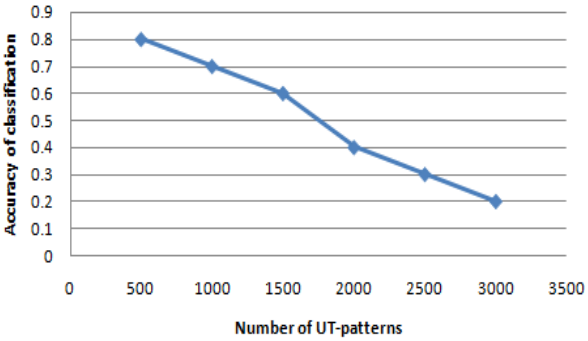

Figure 6 Performance based on number of UT-patterns obtained to the accuracy of classification.

The graph in Figure 6 shows the effect of the number of trajectories per cluster on accuracy. Intuitively, a larger number of trajectories per cluster imply stronger support for the existence of patterns.

When the number of trajectories was about 500, the accuracy was nearly close to 100 percent until number of UT-patterns become more than that. This property of handling larger clusters efficiently is indeed desirable in the big data era. Experiments using real-world spatio-temporal datasets [5] show that UT-pattern mining easily discovers various types of trajectory patterns.

V. CONCLUSION

A framework for mining unifying trajectory pattern based on varying temporal tightness is proposed in this paper. Based on this framework the trajectory pattern mining algorithm called UT-pattern mining is developed. The algorithm first develops sub trajectory clusters and some initial patterns along with reference movement in the first phase. Then a granularity adjustment generates pattern forest by drill down and roll up to discover more patterns in the second phase. Finally the classification and visualization of the trajectory patterns can be done as time restricted, time delayed and time unrestricted patterns efficiently using this framework. Many other trajectory mining techniques can also be further added to improve efficiency of the UT-pattern mining framework.

REFERENCES

[1] Sacharidis, D., K. Patroumpas, M. Terrovitis, V. Kantere, M. Potamias, K. Mouratidis, and T. K. Sellis, On-line discovery of hot motion paths. Proc. 11th Int. Conf. Extending Database Technol., Nantes, France, Pp. 392–403 (2008).

[2] Jeung, H., M. L. Yiu, X. Zhou, C. S. Jensen and H. T. Shen, Discovery of convoys in trajectory databases. Proc. VLDB Endowment **1**(1): 1068–1080 (2008).

[3] http://code.google.com/apis/maps/

[4] http://earth.google.com/

[5] http://www.fs.fed.us/pnw/starkey/data/tables/ is the starkey project link from which the spatio-temporal animal movement datasets can be fetched as data text files.

[6] Han, J., M. Kamber, and J. Pei, Data Mining: Concepts and Techniques, 3rd ed. San Mateo, CA, USA: Morgan Kaufmann, (2011).

[7] Lee, L.G. J. Han and K.Y. Whang, Trajectory clustering: A partition-and-group framework. Proc. ACM SIGMOD Int. Conf. Manag. Data, Beijing, China, Pp. 593–604 (2007).

[8] Lee, L.G., J. Han and X. Li, Trajectory outlier detection: A partition-and-detect framework. Proc. 24th Int. Conf. Data Eng.,Cancun, Mexico, Pp. 140–149 (2008).

[9] Lee, L.G., J. Han, X. Li and H. Gonzalez, TraClass: Trajectory classification using hierarchical region-based and trajectory-based clustering. Proc. VLDB Endowment **1**(1): 1081–1094 (2008).

[10] Lee, L.G., Jiawei Han and Xiaolei Li, A Unifying Framework of Mining Trajectory Patterns of Various Temporal Tightness. IEEE trans. on knowledge and data engineering **27**(6): (2015).

[11] Benkert, M., J. Gudmundsson, F. Hubner, and T. Wolle, Reporting flock patterns. Proc. 14th Eur. Symp. Algorithms, Zurich, Switzerland Pp. 660–671 (2006).

[12] Bakalov, P., M. Hadjieleftheriou, and V.J. Tsotras,Time relaxed spatiotemporal trajectory joins. Proc. 13th ACM Int. Symp. Geograph. Inf. Syst., Bremen, Germany Pp. 182–191 (2005).

[13] Grunwald, P.D., I. J. Myung, and M. A. Pitt, Advances in Minimum Description Length: Theory and Applications. Cambridge, MA, USA: MIT Press (2005).

[14] Kalnis, P., N. Mamoulis and S. Bakiras, On discovering moving clusters in spatio-temporal data. Proc. 9th Int. Symp. Spatial Temporal Databases, Angra dos Reis, Brazil Pp. 364–381 (2005).

[15] Phan Nhat Hai, Pascal Poncelet and Maguelonne Teisseire, GeT Move: An Efficient and Unifying Spatio-Temporal Pattern Mining Algorithm for Moving Objects (2012).

[16] Yan Huang, Liqin Zhang and Pusheng Zhang, A Framework for Mining Sequential Patterns from Spatio-Temporal Event Data Sets. IEEE trans. on knowledge and data engineering **20**(4): 433-448 (2008).

[17] Yu Zheng, Microsoft Research: Trajectory Data Mining- An Overview. ACM Trans. Intell. Syst. Technol. **6**(3): Article 29 Pp. 1-41 (2015).

[18] Li, Z., B. Ding, J. Han and R. Kays, Swarm: Mining relaxed temporal moving object clusters. Proc. VLDB Endowment, vol. **3**(1): 723–734 (2010).

[19] Li, Z., M. Ji, J.G. Lee, L.A. Tang, Y. Yu, J. Han and R. Kays, MoveMine: Mining Moving Object Data for Discovery of Animal Movement Patterns. Proc. ACM SIGMOD. (2010).