Noname manuscript No.
(will be inserted by the editor)

Extracting Ordinal Temporal Trail Clusters in Networks using Symbolic Time Series Analysis

Aparna Gullapalli · Kathleen M. Carley

the date of receipt and acceptance should be inserted later

Abstract Temporal trails generated by agents traveling to various locations at different time epochs are becoming more prevalent in large social networks. We propose an algorithm to intuitively cluster groups of such agent trails from networks. The proposed algorithm is based on modeling each trail as a probabilistic finite state automata (PFSA). The algorithm also allows the specification of the required degree of similarity between the trails by specifying the depth of the PFSA. Hierarchical agglomerative clustering is used to group trails based on their representative PFSA and the locations that they visit. The algorithm was applied to simulated trails and real world network trails obtained from merchant marine ships GPS locations. In both cases it was able to intuitively detect and extract the underlying patterns in the trails and form clusters of similar trails.

Keywords Spatiotemporal networks, Network trails, Time series Analysis, Symbolic Dynamics

1 Introduction

Modern networks are increasingly evolving into large, dynamic, multi-mode entities due to the ease with which data can be computationally recorded and visualized from such diverse scientific fields as economics, geography, biology, epidemiology, organizational studies etc. In response to the richness of the available data, instead

A. Gullapalli · K.M. Carley
CASOS, Institute of Software Research, School of Computer Science, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15223, USA
Tel: +1 412 268 6016, Fax: +1 412 268 1744
E-mail: aparnag@cs.cmu.edu E-mail: kathleen.carley@cs.cmu.edu

of the traditional representation of networks as individual agents (nodes) and the ties in between them [Wasserman and Faust, 1994], current networks consist of multiple types of nodes with multiple classes of node attributes. An interesting feature in many of these networks is the incorporation of network observation time where the network structure and/or agent behavior is viewed at discrete snapshots of time [Carley, 2004]. These spatiotemporal networks are constantly changing and evolving with time. The analysis of the evolution of agent behavior with time at the microscale could potentially lead towards understanding and predicting the behavior of agents and networks at the macro scale.

One possible way to investigate the spatiotemporal nature in networks is to track each agent's location in space across the temporal domain thus leading to a time-ordered sequence of locations visited by the agents. Thus a time series data sequence can be extracted from a spatiotemporal network by relating it to a trail in the temporal network. A trail in a network is a walk (across time) in which all edges are distinct though some nodes may occur more than once. Thus for temporal trails, it is a set of nodes (locations) visited by an agent sequentially across time. In this sense, it is a time ordered sequence, i.e. a sequence of observations taken at different times [Pena et al, 2001]. As an initial step towards understanding the behavior of the agents across space and time, this paper provides a methodology to extract groups of agents with comparable behavioral patterns in the temporal domain. The goal of trail clustering is to identify and extract agents who visit *similar* set of nodes in *similar* patterns thereby leading to *similar* trails. The actual time of visitation as well as the time spent by the agent at each location is not considered in the clustering paradigm; the order in which the locations are visited determines cluster membership. Evaluating the group behaviors of clustered agents in the spatial domain at each time instance based on results of trail clustering can provide insights that would not have been available by analyzing spatial behavior alone.

The time scale may vary for different networks (years, months, days or seconds) and this information is lost when the time series data is generated but an understanding of the changes in the spatial data over time can provide critical insights into the structure, process and shape of the network. This paper deals with time series data analysis within the purview of spatiotemporal networks. An example of spatiotemporal network is found in geospatial networks where the Global Positioning System (GPS) position information about an agent at each time instance is included [Goodchild, 2010]. Typically the analysis of such geospatial networks is restricted to analysis of the agent interactions at each observation time snapshot [Peuquet, 2001] or with the behavior of a single agent over time.

The goal of the trail clustering algorithm presented in this paper is to present a fast, flexible and customizable tool to extract groups of trails that behave similarly across both space and time. The clustering algorithm proposed in this paper begins by modeling each trail as a probabilistic finite state automata. Clustering is performed at two scales; at the macro level the trails are clustered based on the PFSA into coarse clusters. These coarse clusters at the macro level are further refined by performing agglomerative clustering on them. Several customization parameters are available to allow the adaptation of the clustering algorithm to the type of network data being analyzed. In this paper the results of applying the clustering algorithm on a computer generated validation dataset and on GPS coordinates of ship movements in the English channel are presented.

The paper is organized as follows. Section 2 gives the background and motivation for the proposed approach to trail classification. Section 3 provides a brief overview of probabilistic finite state automata (PFSA) followed by the technical details of representing trails as PFSAs. Section 4 then presents the methodology of clustering these PFSAs for recognizing patterns in the temporal trails. Section 5 implements the proposed algorithm on two datasets, one computer simulated and the other obtained from the GPS coordinates of the movements of merchant marine ships. The results of the trail clustering algorithm applied to these two datasets are presented and analyzed. The clustering results are also evaluated against other established clustering approaches. Section 6 concludes with a summary of the proposed trail clustering algorithm and a discussion of future directions of the work presented.

2 Background and Motivation

There is a wealth of literature related to methods for performing time-series data clustering. Excellent overviews of the methods for clustering time series data are provided [Liao, 2005], [Antunes and Oliveira, 2001] and [Roddick and Spiliopoulou, 2002]. In most cases the nature and source of time series data dictates the applicable approach towards analysis. Time series data clustering methodology is also of interest for recognizing similar subsequences of data within the time series. For example in social networks, sequence analysis is used to answer questions regarding similar orders in social and cultural structures [Abbott and Tsay, 2000] as well as visualization of social diffusion patterns in virtual worlds [Börner and Penumarthy, 2003]. In several instances, the temporal trails are used to derive representative networks where the temporal information dictates the network structure. Community detection approaches are then used to find clusters in this flattened network [Cazabet et al, 2012]. The one drawback of this approach is

that the temporal nature of the trails is no longer retained, though it may be useful when dealing with a very large datasets such as those found from social media such as Twitter and Facebook. Clustering of temporal trails can be done in a supervised or unsupervised manner. An example of model based temporal sequence classification which uses apriori supervisory knowledge is instance based learning where each trail can be represented as a set of instances that describe all the trails. A new trail is classified according to its relation to stored instances. This approach to temporal sequence classification has been used for anomaly detection [Lane and Bordley, 1999].

The approaches to clustering of time series data can be broadly divided into three categories: data based, feature based and model based [Liao, 2005]. An example of data based approach towards classification of temporal sequences is found in Hirano and Tsumato [2004] where the authors analyze several different approaches to classifying temporal sequences obtained from the medical domain. The clustering method proposed in this paper is a model based approach since it extracts a probabilistic finite state automata that represents the raw time series data. Clustering is performed on the model and not on the raw data. There exist several approaches to model generation from time series data including ARMA models [Baragona, 2001], hidden Markov models [Li and Biswas, 1999][Wang et al, 2002] and Bayesian models [Ramoni et al, 2002]. These algorithms deal with time series data generated from such diverse fields as robot sensor data, ecology and tool condition monitoring. The probabilistic finite state automata model of a trail is constructed by visualizing each trail as a time ordered series of subsequences. Clustering of time-series data derived from user web sessions using local sequence alignment with dynamic programming is explored by [Poornalatha and Prakash, 2012]. An alternate approach to sequence based clustering is proposed in Assent et al [2008] which is based on extracting all possible lengths of subsequences in a temporal sequence for clustering purposes. Though it is possible to modify the proposed model representation of this paper to consider all possible subsequence lengths this may impose a level of similarity for clustering that may not be practical for agent movements in real networks. The motivation for clustering explored in this paper is to extract agents that exhibit similar behavior in the order of locations that they visit. It is not necessary that they visit these locations at the same time. Agents that are clustered together are done so because their general spatiotemporal behavior is more similar to each other than that exhibited by agents that do not belong to that cluster.

3 Probabilistic finite state automata modeling

The trails that are extracted from spatiotemporal networks consists of a sequence of locations visited by agents in ascending order of time. As with all networks these locations are descriptive, hence, we require a methodology to convert this time series sequence into a numerical feature vector that effectively quantifies the various patterns observable therein. There have been several approaches to time series data analysis which have been customized to the origins of the network dataset in question.

There has been an increased interest in using symbolic dynamics and statistical mechanics approaches towards time-series analysis and modeling [Shalizi et al, 2002; Schmiedekamp et al, 2006]. The D-Markov algorithm, a method of generating PFSA's, has been previously proposed as a simple, computationally fast approach to model the non-linear behavior of dynamical systems. The first step in symbolic time-series analysis is to convert a continuous time-series signal into an equivalent discrete time-series with no significant loss in information. There are several approaches to this conversion process including the use of the wavelet transform [Rajagopalan and Ray, 2006] and the analytical signal space [Subbu and Ray, 2008]. In the discrete time-space, each discrete time-series value corresponds to a symbol, thus allowing the conversion of any time-series data (continuous or discrete) into an equivalent symbol sequence. The symbol sequence is then converted into a PFSA using the D-Markov algorithm. The D-Markov algorithm [Ray, 2004] has generally been used for detecting slowly evolving anomalies by observing the dynamical system over extremely long periods of time. This algorithm has been adapted in the current paper to model short sequences of network trails as PFSA's, for classifying them based on the location visit patterns. The various states and transitions in between these states of the modeled PFSA are dictated by the symbol sequence that is being modeled. The current section will briefly give an overview of the D-Markov machine and a few of its features, followed by its application to modeling trails.

Consider the trail as a time series whose dynamic, time-dependent structure needs to be modeled. Since the trail is equivalent to a discrete time-series, it can be treated as a symbol sequence (without the need for conversion into a symbol sequence) where each symbol of the series represents the node visited at that particular time instance. For the purposes of symbolic time series analysis, in a symbol sequence, the history of a symbol is the sequence of symbols (possibly infinite) preceding it and similarly the future is the sequence of symbols following the current symbol. To reduce the complexity of the analysis the process (that generated the trail) is assumed to be Markovian of length D; that is the current

symbol is affected only by a history of length D. This implies that the current symbol is dependent only on the D symbols immediately preceding it and not the rest. Similarly, the future of each symbol is assumed to be of length one, i.e., the symbol immediately following it in the sequence. This is shown below in Figure 1.

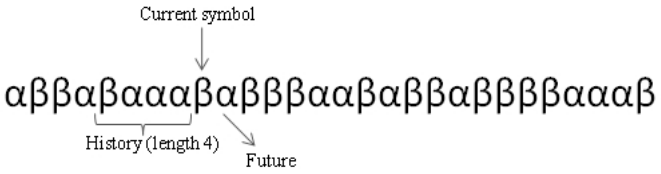

Fig. 1 Time Series with history and future of a symbol

The essential problem that needs to be solved from the modeling perspective is that given the history (of length D) of the current symbol, the future (of length 1) of the current symbol is to be estimated, for each possible set of such histories in the symbol sequence. With this aim in mind, probabilities are assigned to the symbol based on their frequency of occurrence. Then, from a probabilistic point of view, the future can be thought of as the conditional distribution of the future of the current symbol given its history. This conditional distribution is termed as the morph. The PFSA generation problem then is reduced to discovering a partition of the set of histories in the input symbol stream, such that the morph of all the histories in each partition is the same. Thus, the future of all the histories in a partition is the same and each of these partitions is termed as a state of the process.

To provide a computationally fast solution the D-Markov machine defines a known apriori structure. The algorithm assumes that the symbol sequence can be represented as a D^{th}-order Markov chain with an appropriate D. The fixed invariant structure allows for easy pattern representation. A stochastic process X_t is called D^{th} order Markov process if the probability of the current observation depends only on the previous (at most) D observations, i.e., $\forall i$, and $\forall \sigma_i \in \Sigma$.

$$P(\sigma_i/\sigma_{i-1}\sigma_{i-2}...\sigma_{i-D}\sigma_{i-D-1}...\sigma_0) = P(\sigma_i/\sigma_{i-1}\sigma_{i-2}...\sigma_{i-D}) \tag{1}$$

In other words, the process has a memory of length D and hence the system is modeled as a D^{th} order Markov process. Such a process can be represented as a PFSA whose states are defined as "blocks" of length "D" defined over the symbol alphabet, Σ, where "D" is called the "depth". Alternatively, the states can be seen as histories of length "D" in the symbol string. e.g., for a trail consisting of visits to locations α and β only, i.e. an alphabet $\Sigma = \{\alpha, \beta\}$, and depth of two,

the four states are $\{\alpha\alpha, \alpha\beta, \beta\alpha, \beta\beta\}$. For a given symbol sequence and depth, the total number of states in a D-Markov machine is upper bounded by $|\Sigma|^D$. The D-Markov machine structure for depths of one, two and three for a trail consisting of visits to locations α and β is shown in Figure 2.

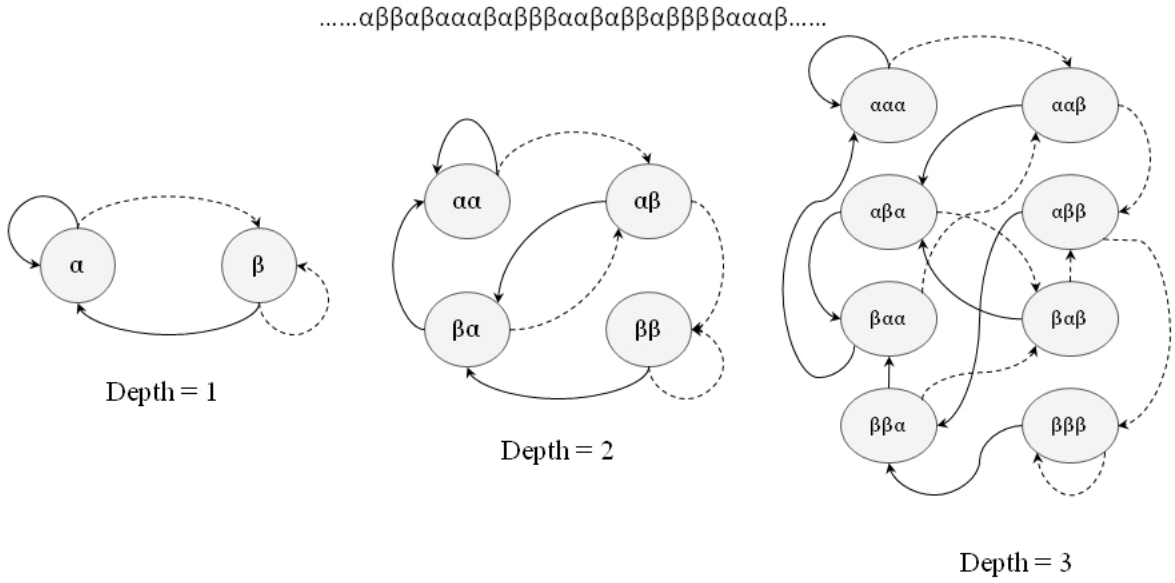

Fig. 2 The structure of the D-Markov machine at different depths for an alphabet $\Sigma = \{\alpha, \beta\}$. The solid lines indicate transitions on symbol α and the dashed lines indicate transitions on symbol β

The transition from one state to the next depends only on the current state and the symbol observed. This probabilistic automaton is termed as a D-Markov machine. The states of the D-Markov and its state transition function δ are fixed as soon as the alphabet size $|\Sigma|$ is selected and the depth of the machine is selected. The only unknown for the probabilistic model then are the transition probabilities, which are calculated from the symbol sequence using a counting measure method. Trails which are similar in nature will have similar frequencies of node visits which will lead to having D-Markov machines with the same structure and similar transition probabilities.

A D-Markov machine can be represented in two ways, namely the state visit probability vector $\mathbf{p}$ and the state transition matrix, Π. $\mathbf{p}$ is a vector of length $|\Sigma|^D$, containing the probability of occurrence of each state in the symbol sequence. However, $\mathbf{p}$ does not contain the complete information about the machine, hence

two D-Markov machines may have the same $\mathbf{p}$, but different Π-matrices. The Π-matrix is a stochastic matrix that gives the probability of transition from one state to another state in the machine. This is a non-negative square matrix with all row sums equal to 1. The π_{ij} element describes the transition probability from state i to state j. For a D-Markov machine with a higher depth D, there will be a large number of zero valued entries and the Π-matrix will be sparse. An alternative way to represent the Π-matrix is with the morph matrix $\widetilde{\Pi}$. The morph matrix $\widetilde{\Pi}$, provides the conditional symbol probabilities for each state. While the Π-matrix is a square matrix, the $\widetilde{\Pi}$ matrix has a size of $|Q| \times |\Sigma|$, where $|Q|$ is the number of states in the D-Markov machine. It follows that for a value of $D = 1$, both these matrices are identical. The state transition matrix for the D-Markov machine of depth two described in Figure 2 is:

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} & 0 & 0 \\ 0 & 0 & \pi_{23} & \pi_{24} \\ \pi_{31} & \pi_{32} & 0 & 0 \\ 0 & 0 & \pi_{43} & \pi_{44} \end{bmatrix}$$

where:
π_{ij}: Probability of transition from i^{th} state to j^{th} state in one hop and state 1 represents $\{\alpha\alpha\}$, state 2 represents $\{\alpha\beta\}$, state 3 represents $\{\beta\alpha\}$ and state 4 represents $\{\beta\beta\}$

$$\widetilde{\Pi} = \begin{bmatrix} \tilde{\pi}_{11} & \tilde{\pi}_{12} \\ \tilde{\pi}_{21} & \tilde{\pi}_{22} \\ \tilde{\pi}_{31} & \tilde{\pi}_{32} \\ \tilde{\pi}_{41} & \tilde{\pi}_{42} \end{bmatrix}$$

where:
$\tilde{\pi}_{ij}$: Probability of occurrence of j^{th} symbol when in i^{th} state, where j= $\{\alpha, \beta\}$.

4 Trail clustering based on PFSA

The modeling of each trail in the spatiotemporal network as a PFSA converts the symbolic sequence of visited locations into a numerical feature vector representing the patterns observed in the trail. The representation of the trail as a numerical vector will allow the usage of a clustering algorithm to classify the trails. Clustering is the partitioning of a data set into groups of similar objects. Representing the data by fewer clusters necessarily loses certain fine details, but achieves simplification in

representation. From a machine learning perspective clusters correspond to hidden patterns, that may not be obvious to a visual inspection.

The aim of the trail clustering algorithm is to find agents which travel similar paths either synchronously or asynchronously. Using an approach like the k-means algorithm which necessitates the apriori specification of the number of required clusters is counterintuitive to this aim. This apriori specification requires some knowledge of the underlying patterns in the trails which is not always possible. In addition, this may also impose a nonexistent structure in the results where trails are incorrectly classified due to the requirement that they belong to one of the prespecified number of clusters. Hierarchical clustering approaches like agglomerative clustering, where each trail is treated as a cluster and these clusters are merged successively based on a distance metric, does not have the problem of apriori specification of the number of clusters. Yet, the problem of choosing appropriate clusters from the hierarchy still remains. Depending on the cutoff parameter chosen this results in dissimilar trails with minimal overlap in the locations visited being grouped into a single cluster, due to a low distance measure between their probability vectors.

Hence this paper proposes a two level approach to clustering of the trails, an initial coarse clustering to partition the input trailset into several groups of trails where all the trails within a group visit similar locations. This is followed by a finer clustering of the previously coarse cluster members to further refine each of these partitions, based on the frequency with which they visit the similar locations. Thus, the clusters of trails are extracted organically from the trail dataset without imposing an apriori limitation on the total number of clusters that need to be formed.

4.1 Coarse clustering algorithm

The coarse clustering algorithm begins with the PFSA using the D-Markov machine algorithm for each of the trails in the input trailset. The length of the probability vector for the PFSA for a trail is based on the depth of the machine and the number of distinct nodes (locations) visited by all the member trails. Thus the size of the alphabet for all the trails is equal to the total number of unique locations visited by all the agents. Though the overall alphabet of the parent trailset may be quite large, a trail need not visit each of the locations specified in the alphabet. The individual alphabet for each trail is determined only by the locations visited by it. This distinction becomes even more crucial for machines with a higher depth, where the number of states grow exponentially with depth.

Coarse clustering partitions the trails in the trailset based on the states they visit. Since the end goal is to extract similar trails, trails that visit the same or almost the same locations are more similar than those that do not visit the same locations. Thus clustering is performed on the trails by grouping trails with the same alphabet into the same cluster. The alphabet of an individual trail is based on only the locations visited by that agent. For example if the parent trailset alphabet is $\Sigma = \{\alpha, \beta, \delta, \varphi, \lambda, \omega\}$, a trail with alphabet $\Sigma_i = \{\alpha, \beta, \lambda\}$ will not belong to the same cluster as a trail with alphabet $\Sigma_j = \{\alpha, \beta\}$ or one with alphabet $\Sigma_k = \{\alpha, \delta, \lambda\}$.

Each trail is represented by its state probability vector $\mathbf{p}$. The Π-matrix or the morph matrix is not used since the memory storage requirements can get quite prohibitive, especially for a large depth value. At the end of the coarse clustering step, the parent trailset is partitioned into several primary clusters based on the locations visited by each trail. In addition, there also may occur some primary clusters which have only one member trail because there are no other trails in the whole trailset which visit all the same locations.

4.2 Approximate clustering

Very often in trail dataset analysis the interest is to look beyond extracting correlated and co-located trails. That is trails that do not visit all the same locations but do visit most of the same locations should be clustered if they visit this subset of same locations with similar frequencies. To extract these interesting patterns amongst the trails the coarse clustering algorithm takes as an optional input the threshold of the similarity percentage amongst the trails. The algorithm then forms coarse clusters by grouping trails with more than N locations in common, where the threshold value N is determined by the user defined percentage. As with the previous approach, these coarse clusters are then refined to determine the final clusters of similar trails. Increasing the depth signifies that trails which have sequences of location visits in common greater than the threshold will get grouped into the same coarse cluster.

4.3 Cluster refining

After the coarse clustering stage the agents have been grouped based on the locations that they have visited. These coarse clusters are further partitioned to extract similar trails within them based on the frequencies with which they visit these locations. It bears repeating here that for a higher depth specification for the PFSA this translates into clustering agents based on the sequence (of length equal

to the depth) of locations that they visit similarly. One of the essential problems in clustering is the determination of the type of distance metric to use for finding similarity amongst trails. Of the possible numerous similarity measures, Keogh and Kasetty [2003] demonstrated that the Eucledian distance outperformed all of them significantly. In addition to the Eucledian metric, the cluster refining was performed using the Kullback-Leibler divergence as well as the angle norm between the probability vectors and no appreciable difference in the performance was observed. Hence by default, the Eucledian distance between the probability vectors $\mathbf{p}$ is used to determine the cluster membership.

The refining algorithm is a radius based clustering approach which begins by constructing the matrix of distances between all the trails within a coarse cluster. The two trails within the primary coarse cluster with the least amount of distance in between them are merged together into a single cluster (if the distance between them falls below a certain threshold). From these two trails, the trail which is located more centrally in the coarse cluster (based on its average distance from the other trails) is chosen as the "archtype" of the cluster against whom all the other remaining trails are compared for membership. The radius of the cluster is determined as a factor of the difference between the minimum distance and maximum distance of the archtype trail with all the other trails in the primary cluster. Once the radius for the cluster is determined all trails in the coarse cluster which are compared with the archtype trail for cluster membership. If the distance between the probability vectors of a member trail of the coarse cluster and the archtype of a cluster is greater than the radius, then that trail forms a separate cluster and becomes the archtype for the new cluster.

Thus at the end of the cluster refining stage, the agglomerative clustering algorithm results in the formation of groups of trails where each of the trails visit the same locations with similar visit probabilities. When the analysis is performed for depth greater than one, this implies that the members in each of the refined clusters visit the same sequence of locations (length of the sequence equals the depth) with the similar visit probabilities.

4.4 The choice of depth

At the end of the trail clustering algorithm the output is a set of clusters. The cluster members are trails with similar state visit probabilities (if the clustering is performed based on the probability vector $\mathbf{p}$). For a specified depth of one, each of the trails in a cluster will have visited the same location with similar frequencies though not necessarily in the same order. If the order in which the locations are visited by the agent is important, then an increase in the depth is recommended.

An increase in depth will allow the grouping of trails based on the frequency of visits to sequences of locations.

One important thing to note in the above discussion is the length of the input trail i.e. the total number of locations visited in a trail. Since the Π-matrix and the $\mathbf{p}$ are stochastic in nature, the length of the trail will determine statistical convergence of their values. Though the algorithm will compute clusters for any depth ($<$ the trail length), at high values, the trails are not sufficiently long enough to ensure that the probability values are statistically significant.

4.5 Related work

Trail clustering is performed by treating each trail as a D-order Markov chain where D is a user specified input that allows the selection of the desired level of granularity in the clustering results. The Bayesian clustering by dynamics [Ramoni et al, 2002] algorithm is similar to our approach where they model each time series data as a Markov chain, except they restrict themselves to using only first order Markov chains. The transition probabilities observed in the Markov chain are used to represent the time series data. Unfortunately due to the restriction of modeling the data as a first order Markov chain any additional patterns in the time series data that occur at a higher order are not discovered. A methodology for discovering modeling time series as a mixed memory Markov model using the Expectation Maximization (EM) procedure is provided in Saul and Jordan [1999]. In contrast, the proposed algorithm is computationally simpler in the model discovery phase while allowing one to retain the advantages of treating the data as a higher order Markov chain. Related work by Smyth [1999] describes a methodology to derive clusters from the Markov chains by partitioning the time series into a predetermined clusters and treating each of them as a mixture model. The EM algorithm is then used to compute the mixture model and to assign time series to clusters to maximize the posterior probability. By contrast, the present algorithm uses the dynamics of the time-series data itself to determine coarse clusters from which agglomerative clusters are generated without the need to apriori specify the expected number of clusters.

5 Empirical results

In this section the trail clustering algorithm is evaluated empirically by applying it to two datasets, one computer generated and the other obtained from the GPS data of travels of merchant marine ships to view the effectiveness of the clustering algorithm.

5.1 Computer simulated temporal trails

To test the accuracy and efficiency of the trail clustering algorithm, a dataset with the trails following certain predefined patterns was generated. This approach will allow us to test if the clustering algorithm can recognize and distinguish between the different patterns.

The test dataset consists of 200 trails. All the trails are of an equal length of 35. Each of the trails in the dataset visits one of two possible locations A or B. The probability of visiting location A and location B determines the "type" of the trail. In the test dataset there are three distinct visit probability distributions to which each trail can belong to. The type of each trail was generated using a random number generator. The dataset contains

- 98 trails belonging to Type 1 where : Prob(A) = 0.5 and Prob(B) = 0.5
- 43 trails belonging to Type 2 where: Prob(A) = 0.75 and Prob(B) = 0.25
- 59 trails belonging to Type 3 where: Prob(A) = 0.25 and Prob(B) = 0.75

A random number generator was used to generate the sequence of locations visited in the trail based on the above distributions. Since a random number generator was used, the distribution of visit locations would be as specified only for very long sequences. For trails that are shorter in length like the ones in the test dataset, the trails will have some variance from the specified distribution values. This also ensures there is sufficient heterogeneity in the dataset to test the clustering algorithm. The temporal aspect of the trail is introduced by treating each visit location occurring at a distinct time period. For the current experiment, the difference between two consecutive time periods is uniform and all the trails begin at the same time period and end at the same time period.

The Loom visualization platform Davis et al [2008] within the ORA analysis platform Carley and Reminga [2004] is used to visually represent all the trails in the dataset. The ORA analysis platform consists of tools and data formats that allow the analysis and representation of all aspects of dynamic networks. Loom uses a waterfall like diagram to visualize the changing nature of a trail over time. Each location in the trail is shown as a vertical strip laid side by side across the viewing area and the transitions between the various locations is displayed by a series of arrows between them. The vertical axis encodes time from the earliest to the latest beginning from the top.

5.1.1 Trail clustering results

The trail clustering algorithm was applied to this dataset with an initial depth of the Markov machine as one. This is the most basic level of pattern classification

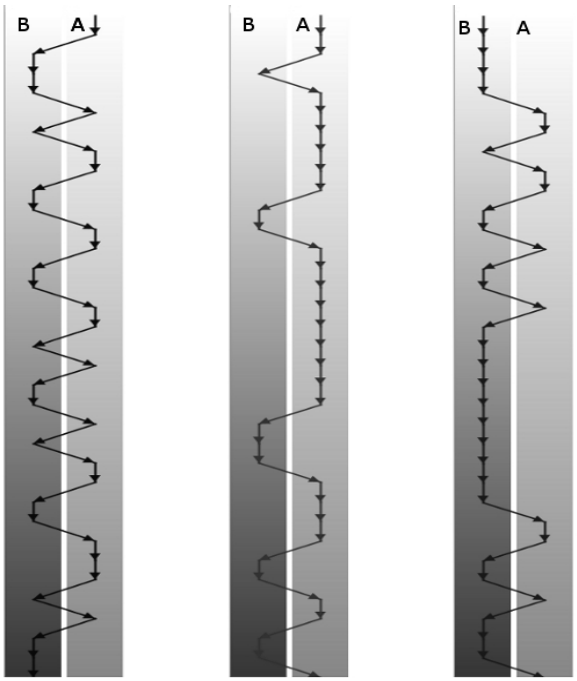

Fig. 3 Examples of the three different types of trails in the test dataset beginning with type 1 on the left, type 2 in the middle and type 3 on the right. The different visit probabilities specified for generation can be seen in the different total number of visits to locations A and B for each of the three trails.

that can be achieved using the algorithm. At the end of the coarse clustering stage all of the 200 trails in the dataset were grouped together into a single cluster. This is expected because all the trails visit the same locations at depth one. The agglomerative clustering refining stage when applied to the coarse cluster resulted in the division of the coarse cluster into three separate clusters, where each clusters archtype belonged to one of the three types of trails in the trailset. Approximate clustering was not performed on this trailset because all the trails visit the same locations. The following table enumerates the results obtained. As seen in column 3 of Table 1, there are 98 trails of type 1 (equal probability of occurrence of A and B) in the original trailset. The trail clustering algorithm clustered 101 trails (column 4) into cluster 1 whose archtype trail belonged to type 1. Of these, 4 trails originally belonging to type 1 got classified into clusters of type 2 and type 3 instead (column 5) and 7 trails belonging to the other types got clustered into cluster 1 (column 6). The reason for this misclassification is that the actual distribution of the visit probabilities though generated from the distribution of {0.5,0.5} were closer to the archtypes for clusters 2 and 3 than the archtype for cluster 1. The

three different trails plotted in Figure 3 are the archtypes for clusters 1, 2 and 3 respectively.

Cluster number	Type of cluster archtype	Original num of trails	Total num of members in cluster	Num of trails not classified	Num of trails misclassified
1	1	98	101	4	7
2	2	43	38	6	1
3	3	59	61	2	4

Table 1 Trail clustering results on the test dataset for a depth of 1

5.1.2 Effect of Varying the Depth

The analysis at a depth of 1 was able to classify the trailset data according the major underlying patterns therein. But the advantage of performing the Markov depth based analysis is that trails can be classified by increasingly stricter conditions of similarity imposed on the sequences of symbols appearing the trails. The test trailset data was analyzed using the trail clustering algorithm at different depths increasing from two to five. For demonstrating the increasing level of similarity in the clustered trails, Table 2 contains the results of the analysis for the members of cluster 2 at depth 1 (row 2 in Table 1). As the depth is increased from 1 to 2, only 33 out of the 38 trails at depth 1 are grouped together in a single cluster. On further increasing the depth to three. 18 of the remaining 33 trails are grouped together into 4 different clusters. At a depth of level 5, which indicates that only trails having subsequences of 5 visit locations in common, only two clusters containing two trails each remain.

Depth	Trail ids of type 1 belonging to same clusters
2	**{4 5 7 16 39 40 43 52 71 80 83 84 85 89 95 98 103 113 118 119 120 121 130 134 136 140 148 161 172 184 185 191}**
3	**{7 113 118 121 130 134 172 184}**, {85 185}, {84 98 120 136 140 148}, {89 95}
4	**{110 113 184}**, {89 95}
5	**{113 184}**, {89 95}

Table 2 Effects of Depth Variation on the test dataset

The trail ids depicted in bold in each row of Table 2 are shown in the Loom waterfall diagram in Figure 4. As can be seen from the progression from the depth of two to a depth of five the trails that are clustered have increasing similarity in the frequency and sequence in which they visit the two locations. Varying the

depth at which the clustering is done allows the user to extract groups of trails based upon the degree of similarity desired. The two trails in the cluster at a depth of five are shown individually in Figure 5. The regions of the trails highlighted by the ovals and square depict the same sequence of locations of length 5 or more (though at different time epochs) visited by both trail 113 and trail 184.

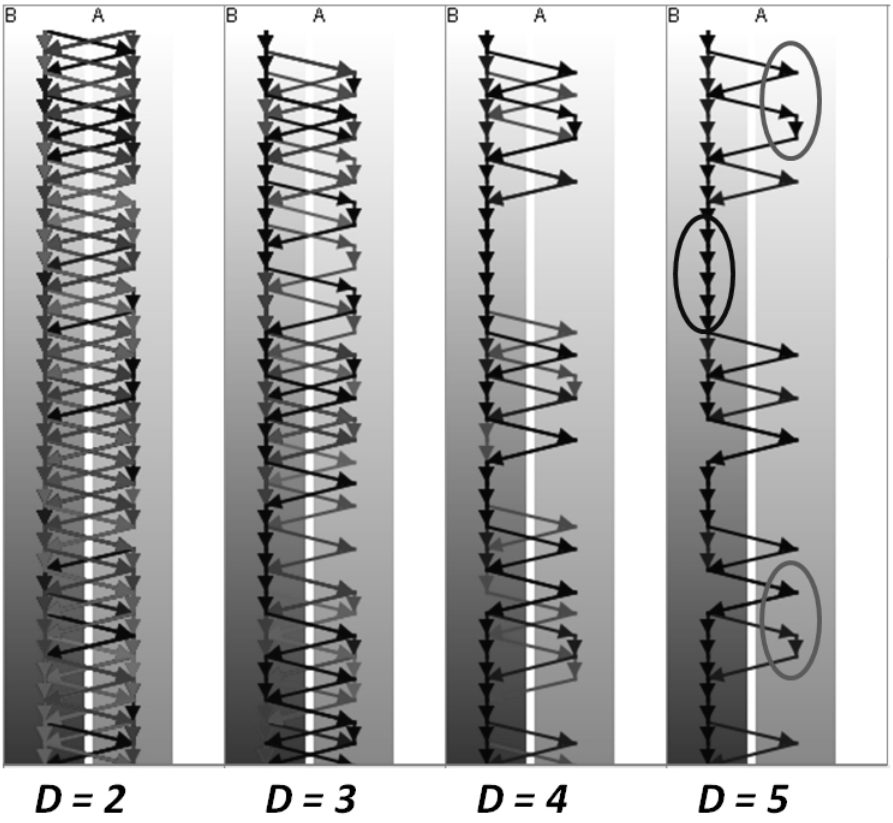

Fig. 4 Loom depiction of the trails in bold in Table 2 at different depths of analysis of the trail clustering algorithm.

5.1.3 Clustering result evaluation

The proposed clustering algorithm's results were evaluated against the agglomerative hierarchical clustering and the k-means clustering approaches. The probability vectors of the PFSAs for each trail were used as the input observations for each of these approaches. The trail clustering algorithm is similar to the agglomerative approach in that both of them start with assigning each trail to a single cluster. The

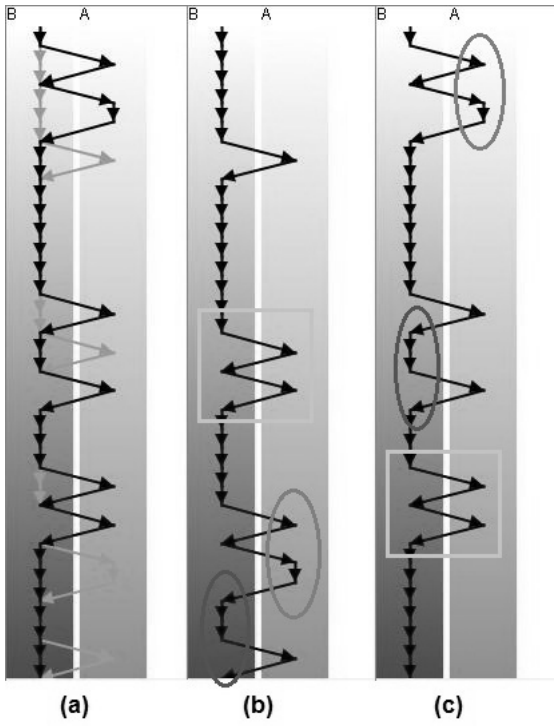

Fig. 5 (a) The cluster at depth 5 as shown in Figure 4 (b & c) The two trails in the cluster, numbered 113 and 184, shown individually. In addition to the long trails of visits at location B, the similar transition between locations A and B of length 5 and more are shown in the circled regions.

merge decisions are based on distances between clusters. Unlike the trail clustering algorithm, agglomerative clustering forms a hierarchical cluster tree from which clusters are determined based on an input cutoff parameter. For evaluation we used the single, complete and Ward's approach to computing distance between clusters. Single linkage computes the distance between clusters as the distance between the two closest elements in the two clusters. Complete linkage computes it as the maximum distance between two elements in the two clusters. Wards linkage computes the distance as the increase in variance for the cluster being merged [Jain and Dubes, 1988]. K-means clustering is a partitional clustering approach which aims to partition the observations into k clusters (specified as an input parameter by the user) in which each observation belongs to the cluster with the nearest mean. In addition to requiring an apriori knowledge of the number of clusters that the data needs to be partitioned into, k-means suffers from the additional drawback that it tends to prefer clusters of approximately similar size. In the test case described above though, we know that the data is derived from three different distributions.

It is encouraging that the proposed algorithm was able to organically partition the data into three different clusters. But we can use this apriori knowledge of the number of classes of the observations to specify as the number of clusters for the k-means algorithm and as the cutoff parameter for the agglomerative clustering trees.

– Internal measure
The internal measure method for comparing clustering results evaluates the clusters based on the data contained within them. The best scores are assigned to algorithms which produce clusters with high similarity within a cluster and low similarity between clusters. The Davies-Bouldin index [Davies and Bouldin, 1979] was used for evaluating the results of each of the clustering results and is computed as follows:

$$DB = \frac{1}{n}\sum_{i=1}^{n} max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right)$$

where n is the number of clusters, c_i is the centroid of cluster i, σ_i is the average distance of all the observations in cluster i to c_i and $d(x, y)$ is the distance between x and y. The Eucledian distance was used to evaluate the distance between centroids. The results are shown below:

Clustering approach	DB Value
Complete Linkage	0.4269
Wards Criterion	0.4669
Single Linkage	0.5930
K-means Clustering	0.4354
Trail Clustering	0.4284

Table 3 Davies-Bouldin index values for clustering results of various clustering approaches

The trail clustering approach had a lower DB value than all of the clustering approaches other than complete linkage agglomerative clustering which marginally outperformed it. Unlike complete linkage, merge decisions for trail clustering are based on distance to a single "'archtype"' observation so it comes at a lower computational cost compared to complete linkage clustering.

– External measure
External cluster evaluation measures can be used to evaluate clustering methodologies when class labels are available apriori for each observation. The clustering algorithm evaluation is based on the class labels of the observations assigned to each cluster. V-measure, an entropy-based external measure was used to evaluate the algorithms [Rosenberg and Hirschberg, 2007]. V-measure

is computed as the harmonic mean of the homogeneity and completeness of the clusters. Homogeneity is satisfied when all clusters contain elements of the same class and completeness is satisfied when all data points that are members of a class are elements of the same cluster. The closer to a value of 1 that the V-measure for a clustering result is, the more homogeneous and complete the clustering results are. Shown below are the V-measure values for the various clustering approaches:

Clustering approach	V-measure value
Complete Linkage	1.0016
Wards Criterion	1.0013
Single Linkage	1.0026
K-means Clustering	1.0012
Trail Clustering	1.0012

Table 4 V-measure values for clustering results of various clustering approaches

The underlying classes for the observations are very distinct and are few in number. This allows the clustering algorithms to arrive at a good partitioning of the observed trails as shown by the relative closeness of the V-measure value to 1. But the trail clustering and k-means algorithm have the lowest observed V-measure value as compared to other clustering approaches. Unlike k-means clustering, no apriori specification of the number of clusters is required by the clustering algorithm presented in this paper.

5.2 Merchant marine ship GPS data

The other dataset analyzed using the trail clustering algorithm was obtained from monitoring the movements of merchant marine vessels in the English channel. From the 25th to 30th of June 2005, a sensor network queried Automated Identification System (AIS) transponders on these vessels, recording navigational details such as current latitude and longitude, heading, speed, reported destination, and several forms of identifying information. In total, movements of over 1700 vessels were recorded, with activities ranging from simple shipping lane traversals to apparently complex itineraries with stops at multiple ports of call. The AIS transponder is directly connected to a Global Positioning System (GPS) and other ship navigational computers, allowing it to automatically generate an accurate report of the vessel's current position.

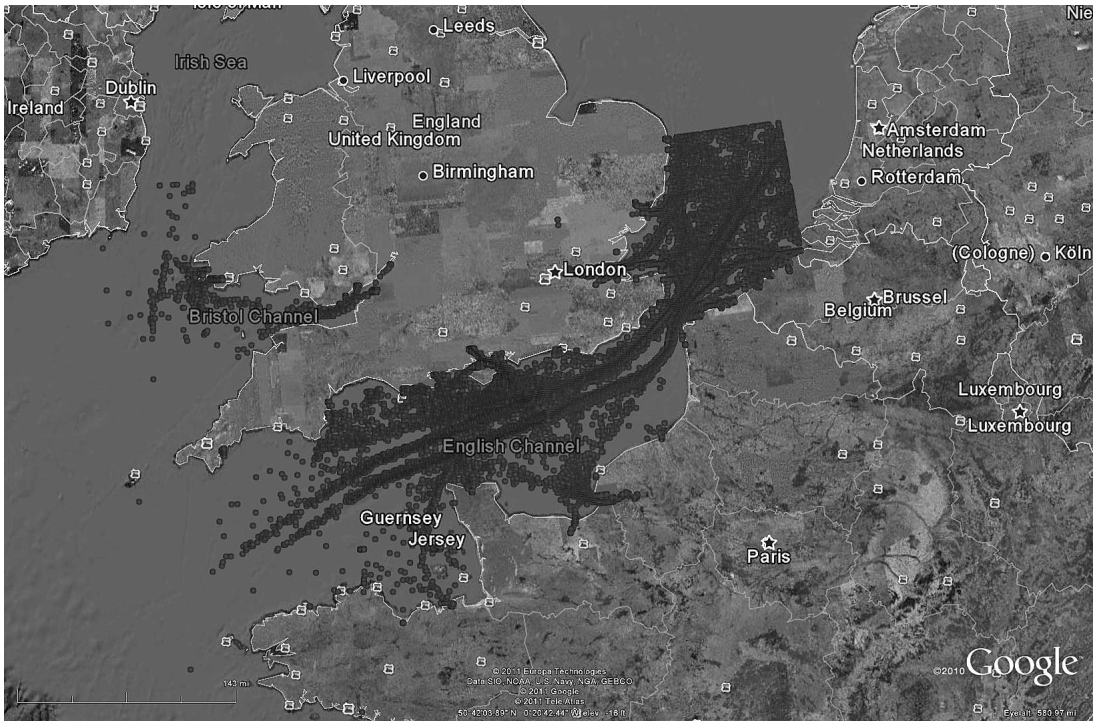

Fig. 6 Geospatial GPS location of the merchant marine ships movement in the English Channel over a period of five days. The GPS locations are displayed using Google Earth

5.2.1 Preprocessing the data

In total the merchant marine dataset analyzed includes 42869 AIS reports from 1729 distinct vessels, over a large geographic range. For the purposes of the current analysis only the latitude and longitude in each of the reports was used to determine the location of each ship at any particular time. Figure 6 shows the locations of all the AIS reports for all the ships based on their geographical positions. In the English Channel area, the effective sensor resolution was approximately 1100 meters, or .6 nautical miles, meaning that small differences in location cannot be accurately distinguished. The trail clustering algorithm presented in this paper treats each trail as time ordered sequence of locations, where each location is represented by a unique id. In contrast, the AIS transponders represent the location as a multidimensional feature vector. The latitude and longitude values were chosen to represent the location of a ship. This also necessitates the conversion of the {latitude,longitude} tuple a single representative (unique) location id. This was done by obtaining the corresponding Cartesian coordinate values on the surface of the earth for each location based on the latitude and longitude values. Each

location is then represented by $\sqrt{x^2 + y^2}$ where x and y are the corresponding Cartesian coordinate value on the surface of the earth. As also stated above the sensor resolution is around 1100 meters. To address this problem with the sensor resolution a distance of 1600m was used to determine the location ids. That is if two locations were within 1600m of each other they were assigned the same location id. This resulted in 217 unique location id's corresponding to the various geographical positions in the input dataset.

5.2.2 Trail clustering results

Trail clustering was applied to the merchant marine dataset using a depth of 1. At depth 1, the total alphabet size for the whole dataset is 217. Due to the diversity of the trails in the dataset and the lack of precision in the obtained geographical positions the approximate clustering algorithm was also used to extract similar trails. The threshold was set to 0.75, i.e. if two trails have more than 75 percent of locations in common they are grouped together into a coarse cluster. One of these trails is picked as the coarse cluster archtype and all further comparisons for similarity are made against this trail. At the end of performing clustering based on visits to the same locations a total of 49 refined clusters containing more than 1 member trail were formed. Out of these, seven clusters had 3 member trails, the rest had two members each. Figure 7 depicts four of the refined clusters in Loom. The corresponding location id for each two-dimensional latitude and longitude coordinates is represented by each of the waterfalls in the Loom diagram.

Figure 7(a) and 7 (b) shows trails which are exactly the same. In Figure 7(a) two of them travel the same path at the same time (they are within 1600m of each other at all times), hence only one of them is visible (the third trail is overlaid by the light gray colored trail). The trail clustering algorithm does not differentiate between similar events that occur at different instances of time as seen in Figure 7(a) and 7 (b). Figure 7(c) shows the results of the agglomerative clustering based on distances between member clusters. The two trails each have different visit probabilities to location 1. But since they visit location 181 with a significantly higher frequency the two trails are grouped together. Similarly, in Figure 7(d) the effects of clustering with a depth of 1 can be observed. The order in which the topmost trail visits the three locations is different from the other two. They get grouped together because the frequency of their visits to these three locations is the same.

The merchant marine ships are not polled at the same locations along their routes. Hence, even if two ships travel the same route they may not be grouped together if all their positional querying locations are separated. In such circumstances approximate clustering is an useful tool. The trailset was clustered further

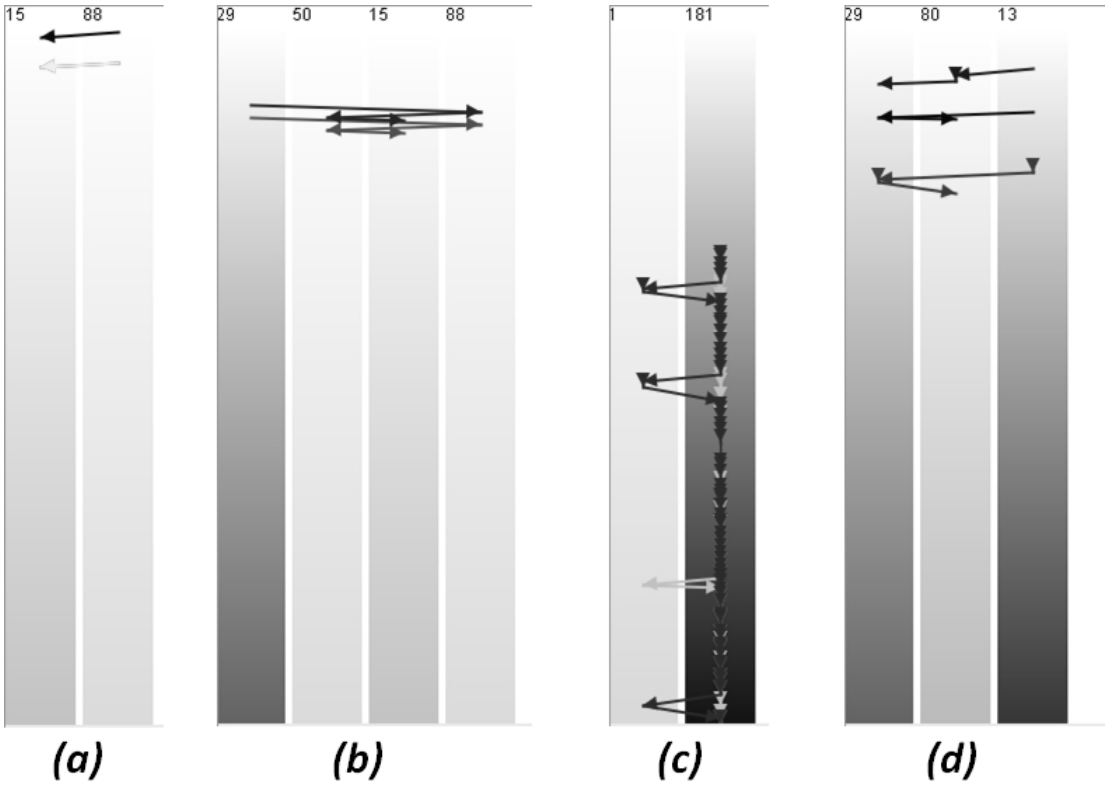

Fig. 7 Loom description of trails belonging to four of the 49 clusters identified by clustering the merchant marine ship GPS data. The trails were clustered based on visiting the same locations only

using approximate clustering and a threshold value of 0.75. This value indicates that if two trails have more than 75% of location ids in common they get grouped into the same coarse cluster. After all the clusters were grouped in this manner, agglomerative clustering of these groups resulted in an additional 68 clusters containing a total of 152 trails. Nine of these clusters are shown in Figure 8.

The benefits of extracting similar trails by approximate clustering can be seen in the cluster depicted in the middle of the bottom row where the gray colored trail is a subset of the black trail essentially following the same geographical path. To further illustrate the approximate clustering the circular trails visualized on the left in the top row of Figure 8 were plotted in Loom. The resulting Loom trails are shown in Figure 9. The circled and boxed regions show regions where there is total overlap between the two trails. This leads to their being grouped together into the same final cluster.

Increasing the depth to two for analyzing the merchant marine ship trails resulted in only 34 trails being grouped together into a total of 15 clusters. Further

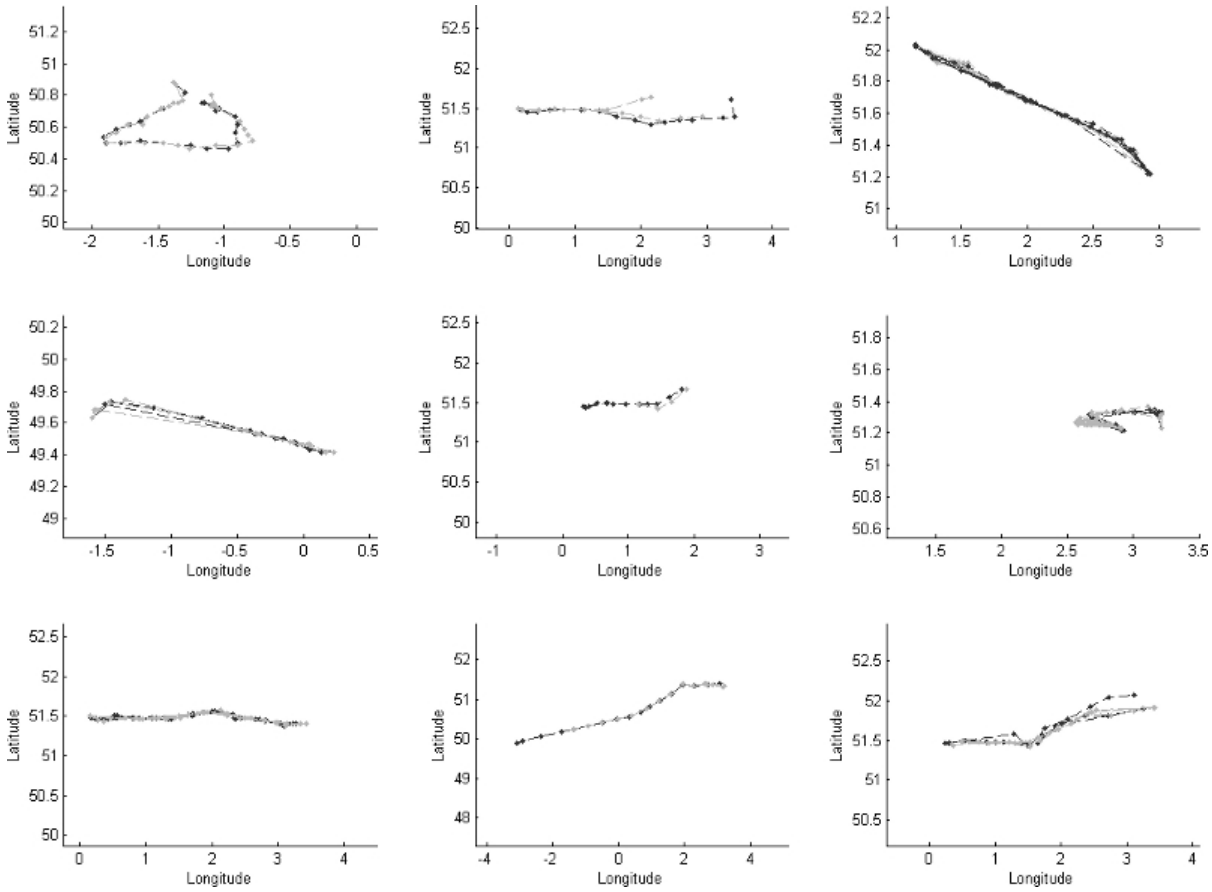

Fig. 8 Nine examples of trail clusters discovered by approximate clustering with a threshold of 0.75 which were not discovered by the regular clustering algorithm. The trails are depicted in the two dimensional latitude-longitude plane. To increase the ease of the visualization of the trails, the axis values of the latitude and longitude in the nine graphs are different

clustering using an approximate clustering threshold of 0.75 resulted in an additional 15 clusters containing 33 trails in total. Thus only 67 out of the initial 1729 trails were classified as part of a cluster; the rest all belonged to one member clusters.

5.2.3 Comparison with other clustering approaches

An external criterion of cluster evaluation like the V-measure cannot be used with the merchant marine ship dataset since apriori class labels are unavailable to compare against the clustered trails. Unlike the simulated dataset where there was some knowledge of the dataset that allowed the estimation of the optimal input for the cutoff values and number of final clusters for the other algorithms,

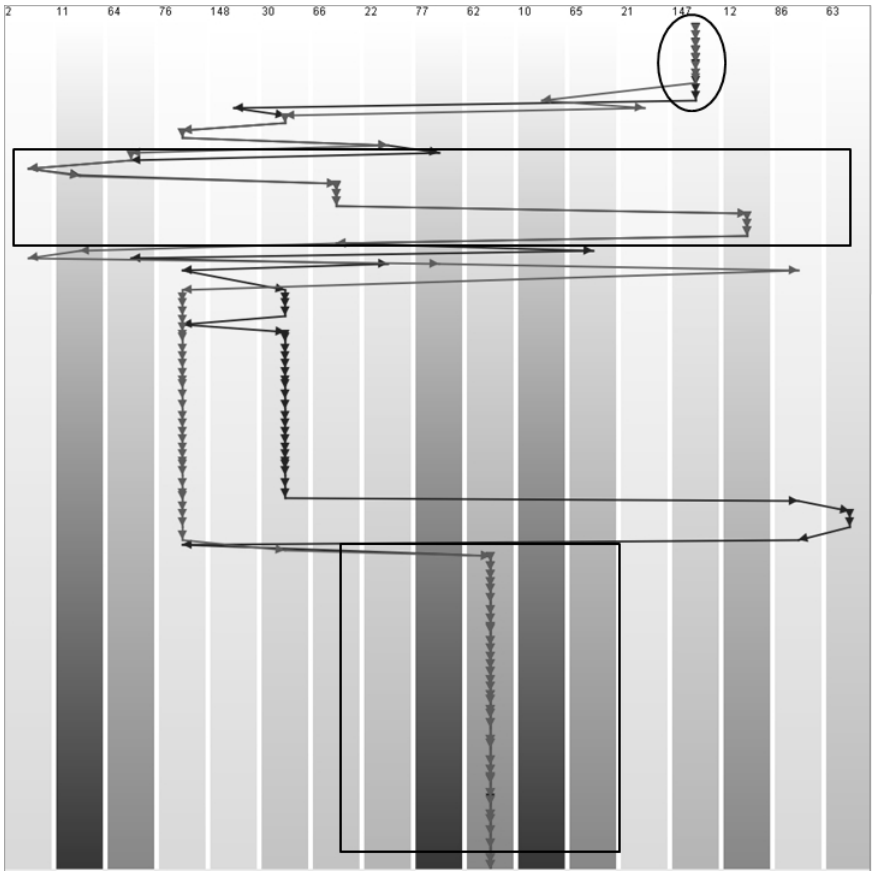

Fig. 9 Loom depiction of the circular trails as seen in the top row of Figure 8.

such information is unavailable here. The trail clustering algorithm resulted in 117 clusters that contained more than 1 member trails. A cutoff value of 200 clusters was used for the agglomerative hierarchical clustering tree methods. The same value of 200 was used for the k-means algorithm as the number of final clusters. The DB values for the various clusters are shown below:

Clustering approach	DB value
Complete Linkage	0.9087
Wards Criterion	1.1999
Single Linkage	0.7357
K-means Clustering	1.2995
Trail Clustering	1.0479

Table 5 Davies-Bouldin index values for clustering results of various clustering approaches for the Merchant Marine ship dataset

Though it appears that the single linkage and complete linkage agglomerative approaches outperform the rest of the clustering algorithms significantly a closer inspection of the resulting clusters shows that very disparate trails are being clustered together in the hierarchical tree. An example cluster for single linkage and complete linkage are shown below in Figures 10 and 11. The advantage of the bi-level approach to clustering is that such disparate trails will rarely get grouped together as similar at the coarse clustering level.

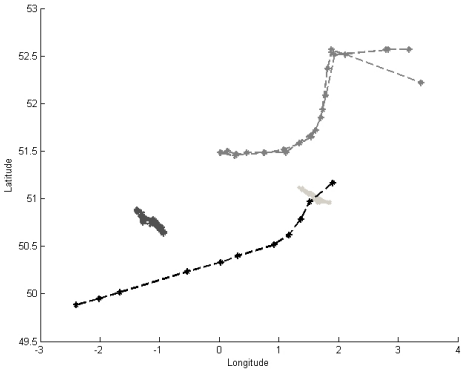

Fig. 10 Example of trails clustered together by Single Linkage agglomerative clustering

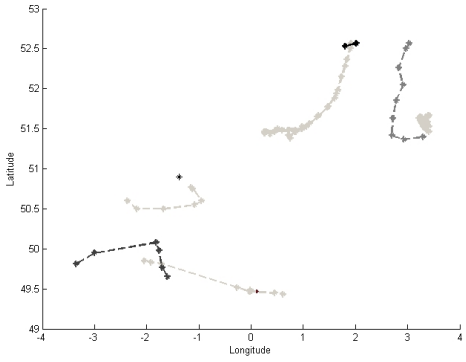

Fig. 11 Example of trails clustered together Complete Linkage agglomerative clustering

6 Conclusions

The main goal of this paper was to provide an approach to clustering spatiotemporal trails in networks. A probabilistic finite state automata (PFSA) approach based on Markov theory was proposed to model each trail. The advantage of using PFSA modeling is that it allows the representation of each trail as a numerical vector. The representative vectors for each of the trails were then clustered together using agglomerative clustering. The advantage of agglomerative clustering is that it does not require the prespecification of the total number required clusters. The use of Markov theory in the PFSA construction allowed the flexibility of fine tuning the level required similarity between trails based on the needs of the user.

The trail clustering algorithm was applied to two trail datasets, one computer simulated and the other obtained from the actual paths taken by merchant marine ships in the English channel. The trail clustering algorithm was able to intuitively cluster the three different trail types in the simulated dataset. It was also reliably

able to derive sets of trails with a higher degree of commonality with increasing depth. The usefulness of the trail clustering algorithm was demonstrated by its ability to parse the large and dense trails in the real world merchant marine ship data and extract meaningful clusters of similar trails and isolate trails that do not behave like any other trail in the dataset. This is especially important in security related applications because the global shipping network plays an important role in smuggling and terrorist attacks. If automated techniques such as the trail clustering algorithm can isolate trails of ships that behave contrary to the overall expected behavior, then targeted surveillance and interventions can be performed. The effectiveness of the approximate clustering algorithm was also demonstrated by the ability of trail clustering to extract clusters of trails that follow the same path but have different polled geographical GPS locations.

The PFSA modeling allows the representation of the trail with the state transition matrix in addition to the probability vector used in the current paper. The state transition matrix for each PFSA contains information in the form of the likelihood of visiting a particular location while in the current location. Since the transition matrix contains more information regarding observed trail patterns it may be worthwhile to investigate if there are improvements in the derived clusters of trails. We did not use the state transition matrix in the current algorithm due to the higher computational memory requirements, especially at higher values for the depth of the PFSA. We are currently investigating different approaches towards adapting the trail clustering algorithm to use the state transition matrix in a computationally efficient manner. The one main drawback of the current algorithm is its insensitivity to time differences. Though the temporal element dictates the sequence in which the locations are visited, the amount of time between two consecutive locations in the trail is not a factor in clustering. For example two trails that visit similar locations with similar frequencies will be grouped together even if these visits occur at different time scales, one in a matter of days and the other across months. We intend to investigate methods to deal with time instance sensitive trails in the future. One potential advantage of applying trail clustering to spatiotemporal networks is the eventual possibility of making decisions or predictions of future behavior for trails based on the PFSA of the cluster. This can be done by using the state probability vector or the state transition matrix to predict the future based on the next location likelihoods of the current state (location/sequence of locations).

7 Acknowledgments

This work was supported in part by the Office of Naval Research (N00014-06-1-0104) for adversarial assessment and (N00014-08-11186) for rapid ethnographic assessment, the Army Research Office and ERDC-TEC (W911NF0710317). Additional support was provided by CASOS - the center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research, the Army Research Institute, the U.S. Army Engineer Research and Development Centers (ERDC), Topographic Engineering Center or the U.S. government.

References

Abbott A, Tsay A (2000) Sequence analysis and optimal matching methods in sociology: Review and prospect. Sociological Methods & Research 29(1):3 – 33.

Antunes CM, Oliveira AL (2001) Temporal data mining : an overview. KDD Workshop on Temporal Data Mining, pp 1 – 15.

Assent I, Krieger R, Glavic B, Seidl T (2008) Clustering multidimensional sequences in spatial and temporal databases, Knowl. Inf. Syst.,16(1):29 – 51.

Baragona R (2001) A simulation study on clustering time series with metaheuristic methods. Quaderni di Statistica, 3.

Börner K, Penumarthy S (2003) Social diffusion patterns in three-dimensional virtual worlds. Information Visualization 2(17):182–198.

Carley KM (2004) Dynamic network analysis, In: Committee on Human Factors, National Research Council, 133 – 145.

Carley KM, Reminga J (2004) Ora: Organizational risk analyzer. Technical Report CMU-ISRI-04-106, Institute for Software Research International, Carnegie Mellon University.

Davies DL, Bouldin DW (1979) A cluster separation measure. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-1(2): 224 – 227.

Davis G, Olson J, Carley KM (2008) OraGIS and loom: Spatial and temporal extensions to the ORA analysis platform. Technical Report CMU-ISR-08-121, Institute for Software Research International, Carnegie Mellon University.

Goodchild MF (2010) Twenty years of progress: Giscience in 2010. Journal of Spatial Information Science 1, 3 – 20.

Hirano S, Tsumoto S (2004) Classiffication of temporal sequences using rough clustering. Fuzzy Information, 2004. Processing NAFIPS '04. IEEE Annual Meeting of the, 2: 711 – 716.

Jain A, Dubes R (1988) Algorithms for Clustering Data. Prentice-Hall, Inc.

Keogh E, Kasetty S (2003) On the need for time series data mining benchmarks: A survey and empirical demonstration. Data Mining and Knowledge Discovery, 7:349 – 371.

Lane T, Brodley CE (1999) Temporal sequence learning and data reduction for anomaly detection. ACM Transactions on Information Systems Security, pp 295–331.

Li C, Biswas G (1999) Temporal pattern generation using hidden markov model based unsupervised classiffication. Advances in Intelligent Data Analysis, volume 1642 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg. pp 245 – 256.

Liao TW (2005) Clustering of time series data – a survey. Pattern Recognition, 38(11):1857 – 1874.

Pena D, Tiao G, Tsay SR (2001) A course in Time Series Analysis, Wiley Series in Probability and Statistics.

Peuquet DJ (2001) Making space for time: Issues in space-time data representation. GeoInformatica 5:11–32.

Poornalatha G, Prakash SR (2012) Web sessions clustering using hybrid sequence alignment measure (HSAM). Social Network Analysis and Mining, 1869-5450, pp 1–12, in press.

Rajagopalan V, Ray A (2006) Symbolic time series analysis via wavelet-based partitioning. Signal Processing, 86(11):3309-3320.

Ramoni M, Sebastiani P, Cohen P (2002) Bayesian clustering by dynamics. Machine Learning, 47:91 – 121.

Ray A (2004) Symbolic dynamic analysis of complex systems for anomaly detection. Signal Processing 84(7):1115 – 1130.

Cazabet R, Takeda H, Hamasaki M, Amblard F (2012) Using dynamic community detection to identify trends in user-generated content. Social Network Analysis and Mining, 2(4):361 – 371.

Roddick JF, Spiliopoulou (2002) A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and Data Engineering, 14:750 – 767.

Rosenberg A, Hirschberg J (2007) V-measure: A conditional entropy-based external cluster evaluation measure. In EMNLP-CoNLL'07, pp 410 – 420.

Saul LK, Jordan MI (1999) Mixed memory markov models: Decomposing complex stochastic processes as mixtures of simpler ones. Machine Learning, 37:75 – 87.

Schmiedekamp M, Subbu A, Phoha S (2006) The clustered causal state algorithm: Efficient pattern discovery for lossy data-compression applications. Computing in Science and Engineering 8(5):59 – 67.

Shalizi CR, Shalizi KL, Crutchfield JP (2002) An algorithm for pattern discovery in time series. Technical Report 02-10-060, Santa Fe Institute, arxiv.org/abs/cs.LG/0210025.

Smyth P (1999) Probabilistic model-based clustering of multivariate and sequential data. In: Proceedings of Artificial Intelligence and Statistics, Morgan Kaufmann, pp 299 – 304.

Subbu A, Ray A (2008) Space partitioning via Hilbert transform for symbolic time series analysis. Applied Physics Letters, 92(8):084107 – 084107-3.

Wang L, Mehrabi MG, Kannatey-Asibu E (2002) Hidden markov model-based tool wear monitoring in turning. Journal of Manufacturing Science and Engineering, 124(3):651 – 658.

Wasserman S, Faust K (1994) Social Network Analysis, Cambridge University Press.