# Algorithm for Cluster History Characterisation and Cluster Association: A graph based approach

**Flinders University**

**Submitted by:**

**Name:** Ashin KC
**FAN:** KC0020
**Supervisor:** Prof. John Roddick

## Declaration

I Ashin KC declare that the thesis report presented for the degree of master's in science (Computer Science) entitled "Algorithm for Cluster History Characterisation and Cluster Association: A graph based approach" has been composed entirely by myself and not been submitted for any other academic degree. The thesis report is solely the result of my own work except where referenced and acknowledged.

## Abstract

The paper proposes new method for the cluster history characterization and finding cluster associations. Iterative searching for clusters over multiple stages in time span is relatively simple, however the knowledge of cluster behavior over time gives temporal characteristics of clusters which serves as a real utility on tracking the cluster evolution history. Similarity coefficient is used as major tool to quantify the cluster evolution path between two successive time intervals. A directed graph model is purposed for the representation of successive progression of cluster over time using similarity coefficient as the weighted edges between nodes. Searching and characterization process of evolving clusters of different instances or objects in a graph provides the frequent substructures and cluster path trajectory which is the significant step towards finding cluster associations.

## Table of Contents

## List of Figure

## List of Tables

## 1. Introduction

Data mining technique has proven to be the most important tool successfully applied on several application areas such as data analytics, machine learning, prediction forecasting and many more. Unlike the traditional definition of data mining over the raw data we make a different approach by analyzing the derived data for further knowledge discovery that are extracted from the data mining process. More specifically, analysis of state of clusters in its life span is the central idea of this paper. Cluster analysis can give us sensitive information about minor changes in data patterns. Such instability and changes in cluster is important factor that can give high-level structure about the data often giving interesting structure of knowledge. Such clustering over time and tracking changes would require clustering taken from the data at different points in time. Clustering over evolving data stream can give us the different clusters at different time points. Spatio-temporal data clustering deals with the location-based attribute at a time for clustering, thus, can result into different clusters. Clustering data with an object described over time series is ever changing and such changing cluster property can be termed as ‘cluster evolution’. Data evolution using cluster analysis has always been studied in various domains however, the study of cluster evolution is gaining interest of many researchers in the recent years.

In this report a graph-based approach to model the cluster evolution tracking is presented. Further, we elaborate the notion of cluster evolution characterisation, understand the importance of various distance metrics and their implications and thus establish a data mining technique from the graph. Cluster evolution is characterised with the temporal behaviour that they exhibit at different time point over the life span of the cluster. Factors like object migration, cluster splits, cluster merging, cluster emerging or disappearing are important characteristics for which similarity distance metric is used to measure such activities of the cluster. Then model is purposed to record these activities into the graph data which proves to be a major tool for the cluster history tracking.

In this paper we describe the information about following as major focus about the research:

- Cluster evolution characterisation
- Mining sequential path evolution of clusters over time
- Cluster history characterisation based on the object migration pattern between clusters
- Graph model to reflect the cluster evolution over time
- Mining technique developed to determine the overall behaviour for analysing changing nature of the clusters.

Further analysis of clusters and their involvement with other dimension like time from the feature space would certainly allow us to relate numerous questions. Most of the data mining algorithms have been used for knowledge discovery directly from the collection of data. However, in recent times, various researches have started to focus on mining rules derived from other mining rules or routines which can also be termed as meta-mining (Roddick and Spiliopoulou, 1999). This method can prove to be significant in cases where the data sets are huge as changes in observations can be detected easily by simply analysing the changes in the extracted rules over time.
Clustering for example, can partition associated objects together but ideal number of partitions, grouping strength among clusters or outlier objects depends on the clustering algorithms, so further analysis is required, that can answer major questions like the commonly associated objects over time, number of stable clusters, frequent patterns of migration of objects, or temporal nature of the cluster that may appear and disappear during the time period. At recent times, the research (Roddick and Spiliopoulou, 1999), have been focused on knowledge extraction directly from the collected data mining results and much focus has shifted towards mining of data with regards to time component. The mining of data where the data itself is the previously mined data can give us the basis for

- Analysis of the mining results
- Development of higher order explanations in describing facts about data, particularly those describing changes over time or location
- Use of higher order data mining algorithms to monitor and describe changes in rule sets.
- Simplification of various knowledge discovery routines
- Reduction of processing time due to the reduced data size,

Analysis of the mining results leads to the development of higher order rules. These higher order rules are useful for analysis of time series and pattern discovery. Thus, incremental discovery of mining rules yields to comparable results that can further describe the evolution of rules. The term higher order mining was introduced by Roddick, et al., (2008). In order to understand the notion of higher order mining it is important to change the perspective about data mining from data analysis to the analysis of patterns, rules or clusters for knowledge discovery. Knowledge discovery has always been based upon large quantities of data such that higher the data spectrum higher the value of knowledge resourced from it. However, sometimes the data pool is not always large enough or the data is not easily accessible. Not all resource institutions share their valuable data and this unavailability of data or the availability of data for a short period time can cause problems in route to knowledge discovery. Hence in many cases data mining is to be conducted without the availability of raw or primary data. Researchers have to analyse through the summarised data that are actually the results of previous data mining techniques. It is a process of analysis of patterns which is beneficial in many aspect of knowledge discovery where very limited data is available or the available data is not the primary data. In addition to that higher order data mining can reduce the efforts to undergo data mining of enormous volumes of data that can be fairly time consuming. In the context of clustering, which is the primary focus of our research, clustering property as described in the literature (Roddick, et al., 2008) can be utilized in various applications. Clustering for association rule discovery adding time factor is harder done than said, because the data is non primary which causes a number of challenges. Another challenge is clustering upon ordinal values such that the relative complexity arises when previous mining algorithms that do not use clustering, making it more difficult to measure the trade-off between time and semantic of primary data. Basic idea here is to use clustering as data pre-processing step to generate rules of sufficient frequency appearing in the cluster evolution pattern. Another aspect of higher order mining is, to cluster the association rules that have been already discovered and formalise them into easy to read and understandable pattern (Roddick, et al., 2008).

Data mining techniques have been increasingly applied to a variety of domains. There has always been a constant need for exploring non-traditional domains for the purpose of knowledge discovery. Data in any domain evolve with time. Network topology, web

graph, protein to protein interactions are few examples of it. Hence, there is an urge for the development of efficient and general-purpose algorithms that are capable of capturing spatial, topological and relational nature of the data set to characterize those domains. Number of researches have been conducted to use graphs to model the complex datasets of various domains. Toxicology is studied using the graph model (Gonzalez, Holder and Cook, 2001) to predict the toxic chemical structure. Several researchers (Dehaspe, Toivonen and King, 1998) have used graph modelling technique to find frequent substructure in chemical compounds. Other example of this is on search space for the image retrieval process (Dupplaw and Lewis, 1999). As supported by various other researchers (Gonzalez, Holder and Cook, 2001; Dehaspe, Toivonen and King, 1998; Chen, C and Yun, 1998) power of graph modelling is applicable to numerous domains and also, graph-based mining is not limited to any particular domain. In the section 3 we review various papers in order to broaden our knowledge to understand the application of cluster history tracking in various domains. The remainder of the report is organized as follows. Section 2 presents the background studies required for the understanding of the problem domain. Section 3 presents the short review of previous works related to our research. Section 4 describes conceptual design and method used for the modelling the algorithm. Section 5 gives the pictorial view of the overall flow of the algorithm. Section 6 describes the algorithm. Section 7 list the pseudocode and section 8 highlight implementation strategy of algorithm. Section 9 puts forward the issues of discussion and finally major conclusions are drawn in section 10.

# 2. Background

## 2.1 Cluster Evolution tracking

The basic objective of clustering is to maximise intra-cluster similarity and minimise inter-cluster similarity. A framework for cluster evolution (Fleder and Padmanabhan, 2006) defines the characteristics of evolution with the help of transition matrix between two different time windows. Typically, the change is defined in three terms; they are characteristic, transitional and associative. The notion of object added/deleted is termed as characteristic change, migration of object is evaluated as a transitional change and lastly, associative change is a measure which gives the degree of

agreement and disagreement such that same objects remain in same clusters or same objects remain in different clusters measured with the help of Rand index (Rand, 1971).

Fleder and Padmanabhan (2006) implemented a loss function where paired measurement is quantified on the basis of distance metric and trade-off between goodness of fit. Similarly, alignment with original clustering is represented with the help of the loss function so as to detect the change. Although this method seems sound and definitive, this research is limited only to a widow of two time slots and thus fails to record cluster evolution over the larger span time of time period which is the primary focus of our research.

Cluster evolution has always been analysed in research (Ramon-Gonen and Gelbard, 2017). This approach is taken to identify similar clusters and detect the migration pattern. An experiment was conducted on a five-year trading data of corporate bond between the year 2010 and 2014. The evolution model hence developed has five stages;

i. Clustering of data per time period,
ii. Identification of cluster similarity,
iii. Determining cluster similarity characteristics at different times
iv. Forming cluster trajectories and
v. Detection of object migration pattern.

However, this research is based on the numerical data points, so the cluster similarity was determined based on distance measure calculated between those numbers. Moving average technique is used to determine the cluster similarity over time and further clustering is done to detect the migration pattern. Clustering over a stream of data can result to different clusters over time and as data input are changing, the resulting clusters are also constantly evolving. Aggarwal, et al., (2003), discuss similar work and propose the idea for dividing the clustering process into online component and offline component. Online component can process periodically and store the detailed summary statistics while offline component can use the statistics to provide quick and wide understanding of data stream and clusters.

Technique for cluster analysis of evolving data streams is devised and furthermore, it is noted that while new clusters are appearing and original clusters are being lost,

there is a shift in position of original clusters which is a crucial factor for the purpose of evolution analysis of clusters (Aggarwal, et al., 2003).

Another approach for tracking the evolving behaviour of clusters over multidimensional records of a data stream is developed and discussed in the paper (Zhou, et al., 2008) in which recent records of the data stream are clustered on the basis of previous record of cluster boundary. Here, cluster boundary is defined in terms of its centroid and it's root mean squared deviation value. Though this approach is able to analyse the evolution behaviour at present window, it falls short in terms of tracking the temporal characteristic of the previous time span. Hopcroft, et al., (2004), take the graph-based approach to detect the community on a large linked network. Basically, periodic agglomerative clustering performed and works on tracking the community evolution over the large network links, give the modern approach to find the temporal evolution structure of community structure (Hopcroft, et al., 2004). The MONIC framework (Spiliopoulou, et al., 2013), is developed for monitoring the changes in cluster with respect to time. The framework intends to identify and overcome the various challenges of monitoring the cluster transition listed as definition of cluster, identifying the cluster at later point of time, what behaviour is the transition behaviour and how to detect those transitions. Furthermore, framework has been tested to detect the community evolution in the social networks, change prediction data streams and trend detection in spatiotemporal data. Ntoutsi, Spiliopoulou and Theodoridis (2011) have presented the model for the effective ways for summarization of cluster changes of an evolving data stream. The graph model is adopted for the representation of cluster transition so that this structure represents the evolution of clusters over time. During summarization process, the less significant transition of these clusters and its information is discarded while producing the compressed representation as the fingerprint of evolution. Moreover, the study (Ntoutsi, Spiliopoulou and Theodoridis, 2011) is focused on the models that adapt to the information loss for the summarization of cluster transition, so that most informative changes are produced from the algorithm.

As discussed previously, we can extend the discussion from the previous studies (Fleder and Padmanabhan, 2006; Ramon-Gonen and Gelbard, 2017; Aggarwal, et al., 2003), that in order to track the cluster history characterising the cluster with respect to its changes is the important aspect.

## 2.2 Cluster history characterisation with similarity measure

As discussed in previous section cluster characterisation is purely a vague topic. To explore what defines the cluster history can be deduced according to the objective and requirements of the research. The list of features that can define and characterise the evolving clusters is hardly complete. However, sensible approach will be to find a method that can accommodate major features to give characteristics of evolution like object migration pattern, repeated patterns, cyclic trends, stability of cluster or instability in other words try to maximise the feature representation.

One of such technique is to measure the similarity between clusters. Cluster similarity is the numerical measure that quantifies, by what value the particular partition is similar or dissimilar with another partition. Various researchers have developed measurement metric for the purpose of comparing partition. Usually there will be two types of data sets. One is data that are in co-ordinate system (number based) and other are in set based (object based). As we will assume that data, we are dealing with are object-based items in a set and will be computing such measure to characterise activity of cluster at that time point. Cosine Similarity measure and Jaccard Similarity measure (pair wise) are good measuring techniques being used in the research. However, for the experiment and comparison purpose we will be using few other similarity measurement techniques. A review of distance metrics and similarity measure in various literatures are listed below.

The semantics of same cluster as successors of previous cluster can be designed according to any of the approach proposed for cluster evolution monitoring (Aggarwal, 2005; Spiliopoulou, et al., 2006). The MONIC approach (Spiliopoulou, et al., 2006) adopts a flexible method which is independent of the clustering algorithm and thus can be used in any type of clusters. In MONIC, cluster succession decision is based on cluster over lap and cluster matching criteria. For example, let c be a cluster from a group of clusters $G_i$ at time t and c' be the cluster from group $G_i$ at t+1 then overlap of c and c' is defined as:

$$\mathrm{Overlap}(\mathrm{C}, \mathrm{C}') = \frac{|C \cap C\prime|}{|C|}$$

According to the Set theory we can visualize the concept of overlap in following figure 2.1.

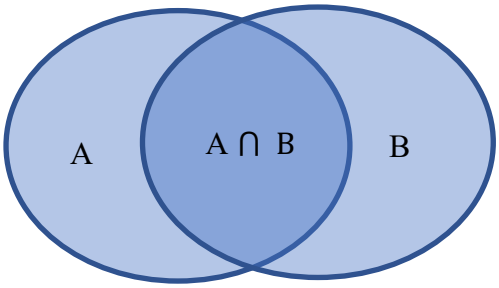


Figure 2.1

Using this overlap concept, Ntoutsi, Spiliopoulou and Theodoridis (2011) developed the evolution graph and characterized the cluster progression properties.
According to these properties, the overlap depends on members of c successively present at t+1, then the best match of c in $G_i$ will be cluster c' that belongs to $G_{i+1}$ .
The maximum overlap to c, or minimum overlap is subject to the threshold. Depending upon the match of old cluster at time t comparing successive new clusters at t+1, transition of old cluster is defined with the following terms.

- *Survival*, $c \in G_i$ survives into $c' \in G_{i+1}$ if c' is the match for c and there is no other cluster in $G_i$ for which c' is a match between time point t and t+1.
- *Absorption*, $c \in G_i$ is absorbed by $c' \in G_{i+1}$ if c' is the match for c and there is at least one more cluster in $G_i$ for which c' is a match between time point t and t+1.
- *Split*, $c \in G_i$ is split into $c_1', c_2', c_3' \ldots c_n' \in G_{i+1}$ with n>1, if overlap of c to each of these clusters exceeds threshold and overlap of all these clusters alltogether exceeds threshold between time point t and t+1.
- *Disappearance*, $c \in G_i$ disappears at i+1 if none of the above-mentioned condition holds true between time point t and t+1.

In the evolution graph, an edge is drawn from c to c' for each of the three types of cases and if the fourth case occurs, there would be no any edge.

Authors (Hopcroft, et al.,2004), conduct the similar work to track changes in large-scale dataset. NEC Cite Seer database, a network over 250,000 papers is clustered

using agglomerative clustering algorithm and change in those clusters is examined in order to track the evolution of cluster over time.

For the clustering purpose similarity between each paper citation reference is associated with the n-dimensional vector $r_n$ and then the similarity between two papers p and q can now be measured as cosine similarity defined by the cosine angle between vector $r_p$ and $r_q$:

$$\text{Cosine}(r_p, r_q) = \frac{rp.\ rq}{||rp||\ .\ ||rq||}$$

Here, numerator is the dot product of two vectors and denominator is the product of length of two vectors. Furthermore, the cluster distance between two clusters C and C' is defined in terms of number of papers in each cluster and their centroid.

$$\text{Distance}(C,C') = \sqrt{\frac{nc.nc'}{nc+nc'}}\,(1 - \cos\,(rc, rc'))$$

Here, $r_c$, $r_{c'}$ are the centroid of the cluster and $n_c$ and $n_{c'}$ is the number of papers in each cluster. To examine the changes in clusters and track the evolution, a modified concept of overlap measure (Ntoutsi, Spiliopoulou and Theodoridis, 2011) is used, if C and C' are two cluster then match value between C and C' is defined as:

$$\text{Match}(C,C') = \min\left(\frac{|C\ \cap C\prime|}{|C|}, \frac{|C\ \cap C\prime|}{|C\prime|}\right)$$

The above formula is derived to deliver the high value of match when two clusters have many papers in common and roughly of the same size. Further, Mohlin (2015) propose the metric for measuring distance between partitions of countably finite set such that it coincides with twice the number of pairs of disagreement and applied to the study of natural language colour categorization.

**Comparison of distance metrices**

A comparative study of distance indices can be found on the research (Denœud and Guénoche, 2006). The paper defines the notion of transfer distance and then various types of distance indices are compared. Rand Index (Rand,1971), Jaccard Index (1908), Rand Index corrected for chance (Hubert and Arabie,1985), Johnson Index and Boorman Index (Arabie and Boorman, 1973) are compared.

Pairwise similarity measure technique is based on the paired agreement and disagreement. Consider x, y ∈ S, two elements are joined in a partition if both belong to same partition, otherwise the elements are said to be separated. Two groups of partition P and Q are said to agree on the pair (x,y) if x and y are joined in both P and Q or x and y are both separated in partition P and Q, otherwise disagree on pair (x, y). Pairwise similarity measure is most applied method for measure of partition in various literatures.

For the further discussion let's consider P and Q be the two groups of partition of n-elements of set S.

$P = \{P_1, P_2, P_3, \ldots. P_n\}$ and

$Q = \{Q_1, Q_2, Q_3, \ldots. Q_n\}$

$P_i$ and $Q_i$ are the number of classes in P and Q. Also, $\theta(P, Q)$ is defined as the minimum number of transfer of elements to turn P into Q established such that maximum numbers of elements in the matched classes that need not to be transferred are kept in same class and keeping account of moving elements. Let's say two elements x and y be a pair of elements. Let *r* be the number pairs simultaneously joined together, *s* be the number of pairs simultaneously separated and *v* otherwise. Also, let $\pi(P)$ be the set of joined pair in P.

*The Rand index*

The Rand index (Rand, 1971), denoted as R is defined in terms of the agreement of two partitions. The index value is in between 0 to 1 range. Thus, similarity measure is number of agreeing pair divided by the total number of pairs. Formally, expressed as:

$$R(P, Q) = \frac{r+s}{n(n-1)/2}$$

*The Jaccard index*

On the contrary to the Rand Index (Rand,1971), in Jaccard Index the pair concurrently joining in their consequent classes *r* is taken into account while s number of pairs separated is left out. Divider to the expression, are the classes of combined elements or total count of combined elements. The Jaccard index (1908), is denoted as J, expressed as:

$$J(P, Q) = \frac{r}{r+u+v}$$

*The corrected Rand index*

Hubert and Arabie (1985) proposed the corrected form of Rand Index. This correction is introduced to anticipate the random chances such that same number of objects are in the same subsequent classes. The corrected Rand index is defined on three values: *r* the number of joining pairs in P and Q, the expected value Exp(r) and the maximum value Max(r), among the classes of P and Q. Maximum value is equal to 1 if the two partitions are equal, expected value is 0 for random chances however, it can also yield negative values.

$$HA(P, Q) = \frac{r - Exp(r)}{Max(r) - Exp(r)}$$

Where,

$$Exp(r) = \frac{|\pi(P) * \pi(Q)|}{n(n-1)/2} \text{ and}$$

$$Max(r) = \frac{1}{2}\left(|\pi(P)| + |\pi(Q)|\right)$$

*The Johnson index*

This index is suggested by Johnson in 1968. The index value is normalization of number of joined pair in both P and Q.

$$J_0(P, Q) = \frac{2r}{\pi(P)+\pi(Q)}$$

*The Boorman index*

This index is not based on the pair of elements joined or separated but simply expressed in terms of the set cardinality, as follows, denoted by B.

$$B(P, Q) = 1 - \frac{|P|+|Q|-2|P \cup Q|}{n-1}$$

Denœud and Guénoche (2006) conducted an experiment in various numbers of elements n, various classes and varying transfer distance. They conclude that Jaccard and Johnson indices give the predicted and correct value to compare close partitions. Further it is noted when number of objects is not too large, transfer distance is

appropriate approach to measure the closeness of partitions conversely it can be stated that transfer pattern can be measured using partition similarity measure.

## 2.3 Frequent substructures in graph

Early algorithm for the frequent item mining was proposed as apriori algorithm by Agrawal and Srikant (1994). Similar approach is taken utilising apriori algorithm applied to the graph data to mine the frequent substructure in the graph (Inokuchi, Washio and Motoda, 2000), where frequent sub-graph is searched in a bottom-up approach while candidates are generated adding one more vertex or edge at a time. However, this candidate generation step is more complex for graph, so frequent substructure mining becomes a harder problem as there will be lot of ways to join two substructures. Authors (Cook and Holder, 2006) point to two major approach for frequent pattern mining in graph dataset, one the apriori-based approach and the other pattern growth approach.

Another solution is to use the similar approach as in frequent pattern tree. In this approach pattern tree or sub-graph is generated recursively and compared to original graph if this sub-structure is the part of original graph. Another interesting substructure mining applied to graph data is, to find topological sub-structures in graph which are frequent enough to be noticed. One primary goal of frequent pattern mining in graph is to discover sub-graphs that occur frequently within the given larger graph dataset.
However, we discuss few terminologies related to the association rule mining to discover how this approach can be applied to find the association rules from graph in our problem domain. Graph based substructure pattern mining ‘gSpan’ (Yan and Han,2002), is well studied and give us important insights about finding the substructure while the topological order is of major concern. Yan and Han, (2002) utilize Depth First Search (DFS) algorithm and construct a search tree to discover all sub-graph without candidate generation and false positive pruning of the main graph. Accommodating the concept of apriori algorithm most frequent edges found in graph are used for the DFS tree construction. Lexicographical order is maintained by using the convention such that vertices form a linear order in the DFS tree, such that if $i<j$ then $v_i$ is discovered before $v_j$ in DFS tree. Vertex $v_0$ is the root while $v_1$ or $v_n$ are the vertex right

to the previous vertices. This form of code is used to represent edges of the graph as 5 tuples of:

$$\{ i , j , L_i , L_{(i , j)}, L_j \}$$

where, $L_i$ and $L_j$ are the label of vertices $v_i$ and $v_j$, $L_{(i,j)}$ is label of the edge between these vertices also referred as the DFS code in the literature (Yan and Han,2002). Furthermore, 'gSpan' algorithm uses isomorphic property of the graph to find the frequently connected sub-graphs by using these DFS code to mine. These DFS codes are used to construct the tree structure in the following manner. Each tree node represents a DFS code, and parent - child relation of the node is well preserved. The relation among the sibling is consistent as described by the isomorphic property of the graph. Using depth, the first search on this tree diagram is used to discover all the frequent sub-graph with DFS codes, conceptually solved as the existing sequential pattern mining algorithms. On the contrary, Ntoutsi, Spiliopoulou and Theodoridis (2011) acquired the similar goal of finding traces of cluster evolution using the graph model, such that cluster transition is recorded in the graph pruning the less informative edges thus sequence of surviving cluster is recorded as a compact summary of evolving cluster. GREW (Kuramochi and Karypis, 2004), is a heuristic algorithm for frequent sub-graph discovery algorithms, designed to operate in large graph dataset and find the connected sub-graphs from a single undirected graph. Algorithm was evaluated using data of different domains like citation linked data, VLSI and web link graphs.

Further we will discuss some useful properties of the substructure in the graph (Cook and Holder, 2006).

*Closed Frequent substructure*

According to the apriori property, subsets of the frequent itemset must also be frequent. Similarly, in graph theory it is derived as all the sub-graphs of a frequent substructure must also be frequent. A frequent pattern is closed if there does not exist a superset of the frequent pattern that has the same support and a frequent pattern is called maximal if it does not have a superset. From this closed property we can infer that the maximal pattern set has the maximum length but closed frequent pattern can reconstruct the whole set of frequent patterns (Cook and Holder, 2006).

*Approximate substructure*

The number of patterns which are slightly similar in structure can be reconstructed to approximate in one such structure however, match problem arises that can discover the variations among these patterns that can produce the approximation (Cook and Holder, 2006).

*Contrast substructure*

Contrast pattern are substructures that are frequent in one set but infrequent in the other set usually compared using minimum support and maximum support. These are useful on finding frequent substructure within the molecular structures as demonstrated in the work of Borgelt and Berthold (2002).

*Coherent substructure*

A frequent sub-graph G is coherent if the information between G and each of its sub-graphs is above some threshold. The number of such coherent substructures in the graph is significantly smaller as a result coherent substructure help to efficiently prune redundant patterns during the mining process. It is applicable in mining where small subset can carry the important feature like distinguishing protein classes from their molecular structure (Huan, et al., 2004)

## 3. Review of Related Works

We further investigate through the review of various research on the application of cluster history analysis.

### 3.1 Comparing Clusters - An Information Based Distance

In the paper (Meilă, 2007) a criterion is formulated that is used to compare two clusters that belongs to the same data set. The criteria known as variation of information (VI) calculates the difference between two clustering by taking account of information loss and gain basically derived from information theory principles. VI is defined as metric and compatibility of VI is examined in different experiential conditions.

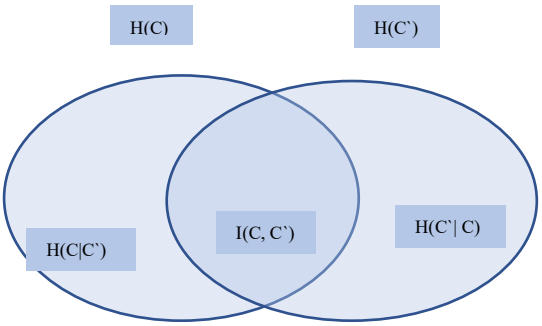


Figure 3.1

If C and C` are two clustering then VI(C,C`) is derived as:

$$VI(C,C`) = 2*H(C,C`)- H(C)-H(C`)$$

Where, H is entropy function, H(C,C`) is the joint entropy, VI consists of following properties:

- VI ( C, C`) is always non-negative
- VI ( C, C`) =0 if C=C`
- VI ( C,C`) = VI (C`,C) are symmetric
- Follows a triangle inequality

From the derivation it is notable that VI is not directedly dependent with the relationships between pairs of points but is based on the relationship between a point and its corresponding cluster in each of two clustering that are being compared.
For *k* defined as the number of clusters in two clustering, the VI distance between two clustering can grow up to the value of 2* log(K). This sets the VI distance different from most of the other metrics which are bounded within 0 and 1 or -1 to 1. Furthermore, for small *k*, no two clustering can be too far apart this is due to the core property of the space of partition that there will be large intersection between clusters of different partitions. From the examples in the paper, this behaviour can be reviewed in the Fowlkes–Mallows(F) and the Jaccard indices(J), with $k = 2$, F is 0.5 and J is 0.13 while with $k = 5$ these values become F = 0.16 and J = 0.02 in contrast that the VI metric yields the significantly separated distance (Meilă, 2007).

In the further discussion authors (Meilă, 2007), make the important notes of their findings that in the practical scenario of comparing clusters, sensible result is acquired while comparing the clusters that are close to each other rather than clustering that are maximally spaced out, so it is rational to assume that clustering are similar to each other. However, it is also clear that their work does not make assumption about the clustering algorithms thus aims to evaluate clustering results irrespective of the underlying algorithm. It has been presented that VI is a metric hence serves as a tool for summarization of different clustering results, however, as it is hard to define a perfect clustering technique with arbitrary context , it is also hard for one to define a comparing criterion that can satisfy every problem optimally. The paper (Meilă, 2007), has attempted to present a logical picture of the properties of the VI criterion, in order to allow a potential user to make informed decisions.

### 3.2 Stability of Market Segmentation with Cluster Analysis – A Methodological Approach

A study (Müller and Hamm, 2014), is focused on the attitude-based market segmentation, using the cluster analysis methodology. The primary goal of a target-oriented marketing strategy to find the distinctive groups of consumers however, the outcomes largely depends on diverse user attitudes and consumer habits. Survey data of food behaviour and preferences gathered from 10,000 German household participating for continuous 4 years was analysed. Results was significant giving valuable information of market segment based on attitudes and consumption habits recorded over given time period. A mixture of various numerical measures was applied so as to ensure objective decision of getting the optimal number of clusters of consumers. The dynamic steadiness of the resulting segments was determined by cluster analyses using data from the same participants in these successive years.

The results show that as the customer preferences is constantly changing so is the market segment so, the challenges arise for designing data analysis process that can well integrate changing consumer preferences as well. A method applied for the data analysis process was to cluster the consumer for market segmentation and thus comparing against clusters from other years. The key fact of market segmentation is to find and characterize specific consumer groups and achieve classification of customers into different segments. The requirements of customers are not essentially

steady and so can cause changes in the marketing pattern, relying on this truth dynamics of objective goals and instability can turn into both beneficial and the curse. As we can see from the research that problem of cluster analysis grows with increasing number of variables. Bias factor comes into play in such procedure during the data sampling steps, nevertheless degree of the biasness was measured in terms socio-demographic criteria it was not hard to exactly determine the influence in final results as data from discontinuing survey participant was not included (Müller and Hamm, 2014).

### 3.3 Mining temporal association rules with frequent item sets tree

The paper (Wang, et al., 2018), suggests a temporal association rule mining algorithm where frequent itemsets tree show temporal relation among numerical time series. Temporal data analysis is growing in popularity as it identifies patterns and regularities among a data set. Temporal association adds temporal dimension to the association rules and thus is helpful to understand the different forms of association rules with respect to time. Wang, et al. (2018) state that trial outcomes show that this procedure can offer better productivity and interpretability in mining temporal association rules as compared to other algorithms and has practical implications in many aspects of data mining. There are various systems involved in temporal data mining like serial association mining, cyclic association mining, stock trading rule mining, patent mining, clinical mining, image time series mining, software adoption and penetration mining, temporal utility mining, fuzzy temporal mining, and calendar association mining (Wang, et al., 2018).

If temporal association rules pass the validations of effectiveness and practicality, they can be used to guide upcoming analytical studies. In the temporal association rule mining temporal and casual relationships among events can be discovered. Temporal association rule mining algorithm introduces the frequent itemsets with temporal constraints though it leads to decreased efficiency because of larger amount of participating items. Several researchers have suggested various methodologies to discover temporal association rule mining including a tree based mining approach and FP tree approach. However these techniques do lack the rules to be less interpretable since they did not regulate the relation between the multi items between inter transactions. The FP-tree based approach and divide and conquer algorithm for

mining multiple time series analyses the time difference between the prerequisite and the consequent rule and thus determines/predicts the trend (Wang, et al., 2018).

To find the frequent temporal association rules, segmentation, discretization and clustering processes are used to convert the numeric time series into the symbolic time series. The main emphasis is on the identification of local patterns in multi variates time series. Moreover segmentation, discretization and clustering of the time series are capable of handling a limited amount of noise by allowing gaps to avoid the overfitting issue due to the noise, local association relation has been utilised (Wang, et al., 2018).

As it is understood by now that temporal reasoning of the past present and future is very crucial for the applications in planning, analysing and diagnosis of a problem. The study indicates that he temporal relations between multi items have huge significance that is proposed by a frequent tree based algorithm. This algorithm can mine the temporal association between the multi items for inter transaction. And also, it can be stored to make more useful decisions by using the FS tree approach. Hence to solve complex problems temporal association rule mining can prove to be an efficient mining tool (Wang, et al., 2018).

### 3.4 Jaccard Index based Availability Prediction in Enterprise Grids

In the recent years, ecommerce, e science has widely gained popularity in a gigantic volume. It has changed the entire mechanism of Science and technology business and also multimedia. Thus, the Grid has emerged as a major distributing system for sharing and also aggregating widely spread data throughout the world. According to their functionality and usability the Grid has been identified as of two major types; global grids and enterprise grids. Global grid as by its name is the grid established over a public network having a global presence. This grid has highly widespread heterogeneous resources. On the other hand, enterprise grid is limited to an institution but spread within it that is, it is accessible only within an enterprise. Hence the enterprise grid is the composition of stored resources that are available to all who are involved with the enterprise. Enterprise grid also known as desktop grid is the accumulation of the resources of the enterprise and shared among its members (Rahman, Hassan and Buyya, 2010).

In an enterprise Grid, this application uses the idle CPU cycles of desktops that are physically dispersed over the enterprise. However, there can be massive computer power that has not been utilised and thus optimising these resources over the internet can deliver high power computing. This can be used in many varied fields such as astronomy, high energy physics, computational biology. SETI@Home, Entropia, XtremWeb and Aneka are some examples of famous enterprise grids (Rahman, Hassan and Buyya, 2010).

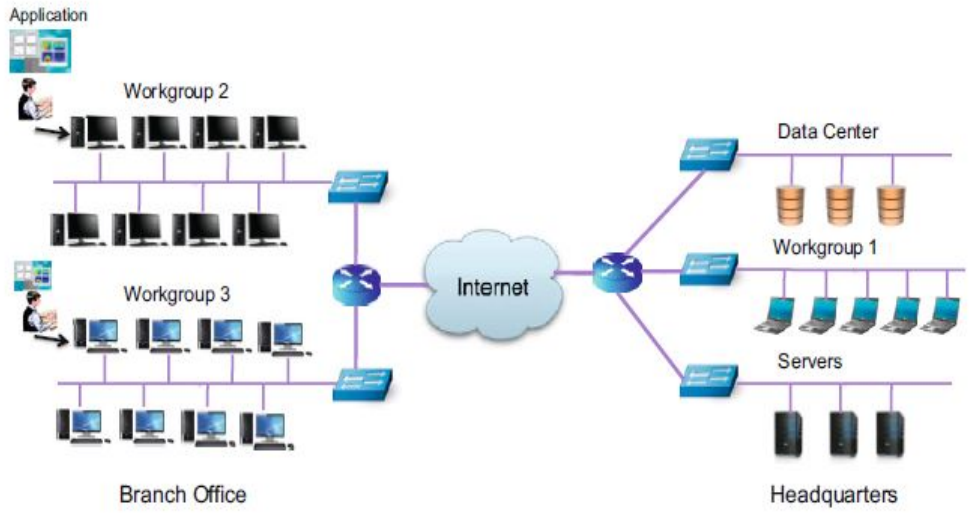


Figure 3.2 Enterprise Grid Architecture (Rahman, Hassan and Buyya, 2010)

Numerous benefits can be extracted from the enterprise grid. It makes the resources simple and easy to access and also non dedicated resources can be utilised. The resources being volatile in nature and also the resources being frequently changing due to its ownership can pose some significant challenges. Thus, unavailability of resources can alter the expected performance of the grid henceforth increases the possibility of task failure in the future. The issues regarding the prediction of machine availability in the enterprise have been thoroughly studied which henceforth help to describe the methodology for predicting this availability of resources distributed over the enterprise (Rahman, Hassan and Buyya, 2010).

One of the methodologies for predicting the availability is the Jaccard Index. In an environment of random and continuously changing data/ resources, it is very difficult to highlight the concrete distribution of the data. In such scenarios, there is a measurement to quantify the degree of similarity and diversity between two data groups. Mathematically this index measures the degree of overlap between two data

sets by calculating the ratio of the number if shares attributed between them. Availability prediction by the use of Jaccard index utilises the comparison of two lazy learning algorithms namely *k*-NN and Naïve Bayes and a hybrid predictor which uses simulation based study. This technique can be further implemented to design availability- aware Grid application scheduling approach. It has practical implications for solving complex scientific applications as well as business application such as workflows related to enterprise grids (Rahman, Hassan and Buyya, 2010).

The main contributions of the work(Rahman, Hassan and Buyya, 2010) include:

- Preprocessing data collected from two enterprise Grid computing atmospheres.
- Put forward Jaccard Index based unique resource obtainability prediction method for enterprise or desktop Grid systems.
- Accompanying a wide simulation based work to show the efficiency of the projected method by means of real world traces.
- Contrast of the suggested method in contradiction of three other renowned obtainability prediction procedures.

Hence, it is found that Jaccard index based prediction methodology results to better and accurate prediction by reducing the computational complexity than in other similar techniques (Rahman, Hassan and Buyya, 2010). Enterprise grid is the sharing and aggregation of resources available within an enterprise which has a wide interconnectivity but posses volatility. Hence it becomes crucial to be able to assess the predictability of the resources to achieve greater performance efficiency and reduce possible runtime failures. There arises a need for a prediction approach that gives higher accuracy and at the same time reduces computational complexity. Here the Jaccard index proves to be a pivotal tool for availability prediction (Rahman, Hassan and Buyya, 2010).

### 3.5 Finding Frequent Trajectories by Clustering and Sequential Pattern Mining

Data mining is a dominant and evolving technology developed to mine summarising information from a giant size of past data. It is helpful to find the recurrent trajectories of moving items in spatio-temporal data and thereby embrace the perceptions of clustering and sequential pattern mining. The researchers have used the procedures that rationally divided the trajectory period area into clusters and then applied the k-

means algorithm over the clusters to thus reduce the squared error. Furthermore, the threshold to acquire active clusters has been applied and they have been placed in descending order created on number of trajectories passing over. From these active clusters, inter cluster patterns are originating by a sequential pattern mining technique. The procedure is recurrent until all the active clusters are connected. The clusters thus associated in sequence are the common trajectories (Shaw and Gopalan, 2014).

With the widespread use and popularity of digital technologies in almost every aspect of modern life there is the availability of huge volumes of data that can be related to the concept of data trajectory collected from various devices. Data collected from communication equipment like, GPS, GSM and logs are few examples among many. The frequent trajectory (FT) patterns from these data may offer the information on the overall cumulative performance of a population of moving objects. This might aid finding substitute trails in congestion control. Hence finding FT assumes high importance and it can be applied with some additional statistical ideas to predict the next level of information in the time series domain (Shaw and Gopalan, 2014).

This approach is highly applicable in electronic communication networks and also road traffic networks. There can be high use so as to find the most frequent path used where the maximum traffic occurs. Thus, based on these finding there can be numerous other practical applications such as finding the best alternative paths in the cases of network congestion. Similarly, the migratory paths of migratory birds can be identified and consequently can be highly useful in managing air traffic for aviation industry. Certain statistical methods like curve fitting techniques and time series trend analysis can be helpful to predict share prices (Shaw and Gopalan, 2014).

# 4. Design and Methodology

## 4.1 Conceptual Design

Clustering over a given dataset would result the list of assigned objects to different clusters. For datasets which are generated from same source over time or similar source having time stamp or time signature generally would be referred as the evolving dataset. Such datasets are of various nature, like at one period of time total objects in dataset may be increasing, while at other time size of same dataset may be decreasing, or say that new data objects appear and disappear over the time period of dataset. Clustering over such dataset at separate time stamp would result the list of

clusters with the time signature which can be used to distinctly isolate the clusters and their objects at different instant of time. Clustering algorithms is not in the scope of this study which would be used to generate resulting time stamped lists of clusters, however resulting time stamped list of clusters and its assigned object are the primary inputs for the purpose of our study.

Tracking the cluster over time also gives rise to problem space of cluster identification, how would we determine that particular cluster is the same cluster which was present at previous time or is the new cluster is just appearing. One approach to solve such identity problem would be to put a label on the cluster so that it can be identified from their label and its evolution path can be tracked. However, this approach would largely depend on the previous clustering algorithm if it can label those clusters otherwise pre-processing would be required to label these cluster before it is taken as input to the algorithm. Other approach would be generating and assigning the identification of such cluster based on their history and membership of the objects at present during the filtering stage. This process of giving identity is a reasonable approach as it would not depend on clustering algorithm, moreover, adds flexibility as criteria of identity constrains would change according to objective of study so it can be manipulated within the algorithm.

Let's formulate representation of given population of objects and the cluster history.

The complete population of clusters can be expressed as:

$$P = \{ (C_1, C_2, \ldots\ldots C_n), \Lambda_n \} \quad (1)$$

Here,

$\Lambda_n$ is the evolution history of entire population. Evolution history is recorded with the help of the Directed Acyclic Graph such that progression of the objects and their characteristics are recorded in the direction of evolving time.

$C_i$ is a cluster defined by its attributes:

$$C_i = ( L_{Ci}, \Theta_{Ci}, \Lambda_{Ci} ) \quad (2)$$

Where,

$L_{ci}$ is Label

$\Theta_{Ci}$ is cluster history and

$\Lambda_{Ci}$ its lineage path such that $\Lambda_{Ci} \subseteq \Lambda_n$

Successive time interval can be defined in terms of their absolute end points, start time and end time such that $t_{s-e} = [t_s, t_e]$

$t_s$ = start time.

$t_e$ = end time.

$\Lambda_C$ is a lineage graph which is the subset of $\Lambda_n$ where $\Lambda_n$ is complete DAG of the population where node of the graph is the cluster at any interval $t_s$ and $t_e$ constructed in following manner.

Clusters that belong to particular time interval $t_{s-e}$ are added as a node to the graph at same depth. Then clusters that belong to successive interval of that instance are added making previously added node as their ancestors. All the cluster which belong to particular time stamp have the same depth in graph and their successive nodes are added in similar manner to $\Lambda_n$ to construct the complete DAG of the population.

Formally,

Lineage path of given cluster population $C_p$ is expressed as:

$$\Lambda_C = \{N, E, S_s, E_e\} \quad (3)$$

Where,

N is the set of nodes

E is the set of edges

$S_s$ is start node

$E_e$ is set of end nodes or leaf nodes

Each node in the graph correspond to the cluster, identified with cluster Label and Time point

### 4.2 Characterisation

Pairwise comparison is well adapted method to compare two partitions based on their agreement and disagreement as a member of the partition. Early paper Rand index (Rand, 1971) is used to measure the quality of the partitions of any set, other popular measure is Jaccard Similarity index first used at 1908's which used pairwise comparison of two groups to calculate the similarity coefficient between two partitions. Cluster similarity is the measure to compute the metric value by which a particular portion is similar or dissimilar with other partition. Here we will be dealing with set of data and we will be computing such factor to characterize the activity of cluster at that point. Cluster behaviour can be summarized in reference with object migrating

between clusters, finding outlier objects, temporal duration of patterns in those data and metric of those changes.

Pair wise similarity technique is used for the similarity measurement. When we treat each cluster as the set of objects and measuring Jaccard similarity in those cluster would yield the coefficient between 0 to 1 which 0 being least similar and 1 being the maximum similarity. Cosine Distance and Jaccard Similarity measure are suitable measuring techniques which record the object migration from one cluster to another. Characterisation step involves the step of comparing two clusters at two successive time points.

For the illustration please consider example below. For the given cluster record at two time point

| Time | Clusters |
|---|---|
| **T1** | C1(a,b,c,d); C2(e,f,g,h); C3(I,j,k) |
| **T2** | C1(a,b,c); C2(d,e,f,g); C3(h,I,j,k) |
| **T3** | C1(a,b,c,e); C2(d,f,g); C3(h,I,j,k);C4() |

Table 4.1

Jaccard Similarity between any two given set is measured in terms of their cardinality as a ratio of cardinality of set from intersection to cardinality of set from union. Mathematically,

$$J_{sim} = \frac{|A \cap B|}{|A \cup B|}$$

Cluster similarity measure using Jaccard coefficient can be shown as

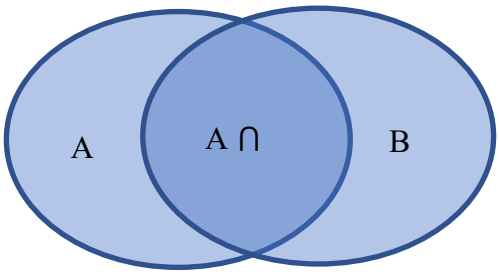


Figure 4.1

For Interval $T_1$ to $T_2$

$J_{sim}(C_1, C_1) = |\{a,b,c,d\} \cap \{a,b,c\}| / |\{a,b,c,d\} \cup \{a,b,c\}| = 0.75$

Similarly,

$J_{sim}(C_1, C_2) = 0.14$

$J_{sim}(C_1, C_3) = 0$

$J_{sim}(C_2, C_1) = 0$

$J_{sim}(C_2, C_2) = 0.6$

$J_{sim}(C_2, C_3) = 0.14$

$J_{sim}(C_3, C_1) = 0$

$J_{sim}(C_3, C_2) = 0$

$J_{sim}(C_3, C_3) = 0.75$

For Interval $T_2$ to $T_3$

$J_{sim}(C_1, C_1) = 0.75$

$J_{sim}(C_1, C_2) = 0$

$J_{sim}(C_1, C_3) = 0$

$J_{sim}(C_2, C_1) = 0.14$

$J_{sim}(C_2, C_2) = 0.75$

$J_{sim}(C_2, C_3) = 0$

$J_{sim}(C_3, C_1) = 0$

$J_{sim}(C_3, C_2) = 0$

$J_{sim}(C_3, C_3) = 1$

For the above calculated value example graph constructed will be of the following form as shown in the figure 4.2

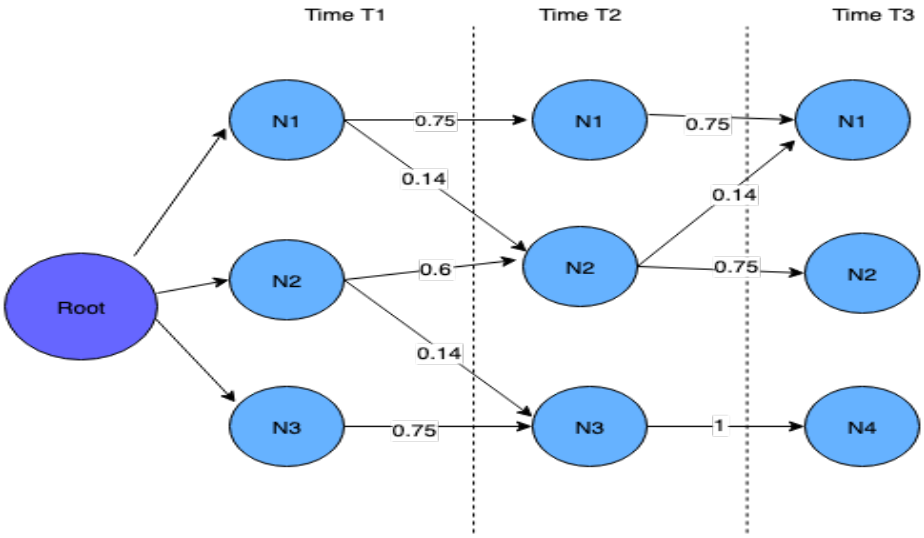


Figure 4.2

**Semantic of Edge**

The edge $e = (n_{i,t}, n_{i,t+1}) \in E$ denotes that cluster $C_{i,t}$ at time t is related to the cluster $C_{i,t+1}$ at successive time t+1. Each edge is weighted, and this weight value is the metric value calculated between the nodes, in the algorithm we are implementing the similarity coefficient as the weight. Weight given to the edge of the graph plays the important role defined such a way that there exists the edge between the two nodes if the weight is greater than the predetermined threshold value otherwise no edge is defined. This weight of the edge quantifies information flow and various characteristics between two nodes. For example, if we use Jaccard similarity coefficient this index can indicate number of object migration from one cluster to another cluster between two intervals of time. However, migrating objects can also be recorded as a crucial information during the cluster comparison process.

**Semantic of Node**

The node $n_{i,t} \in N$ corresponds to the $i^{th}$ cluster found at time t. Node has the same label as cluster moreover, identity of each node is the function of the node label and the timepoint together. The Node in the other hand keeps the track of its ancestor nodes by keeping the record of path followed by its incoming edges. Each node records the list of paths and their count relative to their direct ancestors as a result node is able to store the local trends and temporal pattern occurring till that particular point of time.

Properties of the Node and Edge of the graph would serve as significant feature for the algorithm. Total Number of nodes in graph is the total number of clusters in the given time interval such that Cluster1 at time t1 and Cluster1 at time t2 are two different successive nodes. The lineage graph LG = G(N, E) spans over the period of observation. Set of nodes N corresponds to the set of clusters observed during the time period t. The set of edges $e = (n_i, n_{i+1}) \in E$, is defined such that $1 \leq i \leq t$. Cluster activity is characterised with the help of these edges and pattern progression is recorded at each node thus giving us the structure and major tool to find the frequent structure in the graph.

Conceptual structure Cluster Lineage graph (DAG) of expression (3) is shown in figure 4.3

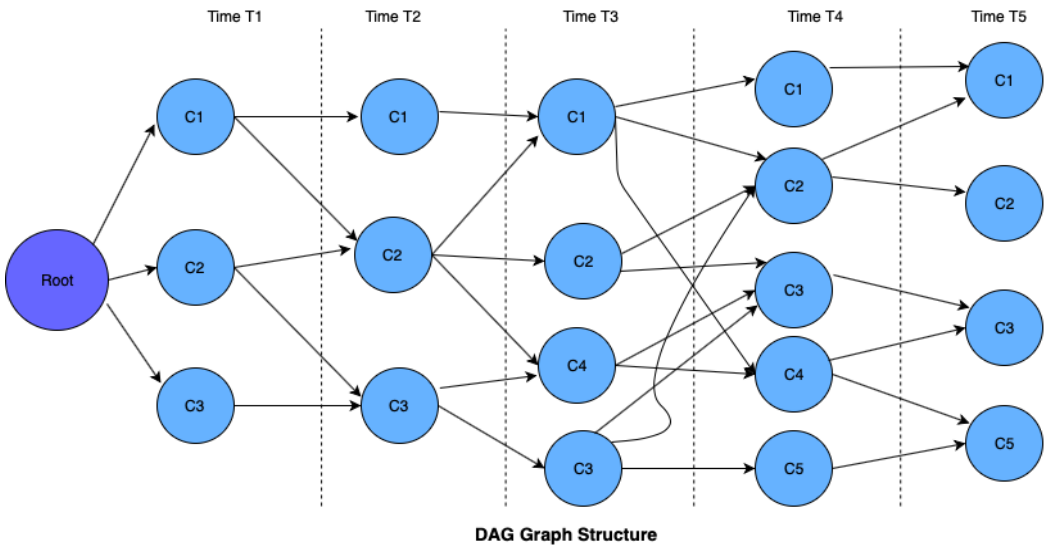


Figure 4.3

### 4.3 Graph Mining to find Association

Graph construction step has been discussed in the previous section. It is important to note that at each graph node instance that it keeps the record of its ancestor path. Each individual node from the expression (3) can be formally described as:

$$N_{i,t} = (E_{i,t} , L_{i,t} , I_{i,t}) \quad (4)$$

Where,

$N_{i,t}$ is the specific $i^{th}$ node at time point t

$E_{i,t}$ is the list of edges

$L_{i,t}$ is the list path followed by its immediate ancestors

$I_{i,t}$ is the identity(Label) of the node

Also, each Edge can be formally described as:

$$E_{Ni,t} = (N_{t-1}, W) \quad (5)$$

Where,

$E_{Ni,t}$ is any edge associated with $N_{i,t}$

$N_{t-1}$ is the parent node associated with the edge.

W is the weight value of edge.

Maintaining the topological order in our subsequence is the important since we are dealing with modified form of traditional graph. As we can note that each node in the graph are unique and described by its corresponding cluster label and its time signature however in the record keeping process, we keep track of the subgraph subsequence which is mainly described by their corresponding cluster label. So, the sequence of path would have same cluster label multiple times in the same substructure. This repetition of cluster label should not introduce any confusion since it is the path or substructure within the original graph, also topological order of these subsequence is very important and are the main findings of the study for the frequent pattern seen in the data. For example, refer to the table 5.2, subsequence $C_2,C_1$ and $C_1,C_2$ are not same and since we maintain the topological order we distinctly identify them as two different sub structure of the graph.

The sequence of path found in a graph, if a node contributes to the support of multiple sequence { $N_{i,1}$, $N_{i,2}$, $N_{i,3}$ ….$N_{i,t}$} of increasing size then node $N_{i,t}$ is a subgraph of $N_{i,t+1}$. Also, for any node we do increase the node count, we do not update record of transition two consecutive nodes of same Cluster Label. Nodes $N_{i,t}$ and $N_{i,t+1}$ are the two consecutive nodes corresponding to same cluster label. Another, important aspect of these sequence is to find their duration in the cluster lineage graph but as our model predicts finding these durations is the obvious answer form graph traversal step. Moreover, it is also important to propagate and assign the metric value of this evolution path, for this purpose arithmetic mean or geometric mean can be used to assign the average coefficient to the particular sequence path based on its previous assignment and present edge weight. Geometric mean of the weights is propagated as overall weightage of the metric.

| Node at t=4 | Sequence structure (pattern → count) |
|---|---|
| $N_{1,4}$ | $C_1 \rightarrow 4$; $C_2,C_1 \rightarrow 1$; $C_1,C_2,C_1 \rightarrow 1$ |
| $N_{2,4}$ | $C_2 \rightarrow 4$; $C_1,C_2 \rightarrow 2$; $C_3,C_2 \rightarrow 1$; $C_2,C_3,C_2 \rightarrow 1$ |
| $N_{3,4}$ | $C_3 \rightarrow 4$; $C_2,C_3 \rightarrow 2$; $C_1,C_2,C_3 \rightarrow 1$; $C_1,C_2,C_4,C_3 \rightarrow 1$; $C_2,C_3,C_4,C_3 \rightarrow 1$ |
| $N_{4,4}$ | $C_4 \rightarrow 2$; $C_1,C_4 \rightarrow 1$; $C_3,C_4 \rightarrow 1$; $C_1,C_2,C_4 \rightarrow 1$; $C_2,C_1,C_4 \rightarrow 1$ |
| $N_{4,5}$ | $C_5 \rightarrow 1$; $C_3,C_5 \rightarrow 1$; $C_2,C_3,C_5 \rightarrow 1$ |

Table 4.2

| Node at t=5 | Sequence structure (pattern → count) |
|---|---|
| $N_{1,5}$ | $C_1 \rightarrow 5$; $C_2,C_1 \rightarrow 2$; $C_1,C_2 \rightarrow 2$; $C_2,C_1,C_2 \rightarrow 1$; $C_1,C_2,C_1 \rightarrow 1$; $C_3,C_2,C_1 \rightarrow 1$; $C_1,C_2,C_1,C_2 \rightarrow 1$; $C_2,C_3,C_2,C_1 \rightarrow 1$ |
| $N_{2,5}$ | $C_2 \rightarrow 5$; $C_1,C_2 \rightarrow 2$; $C_2,C_3 \rightarrow 1$; $C_2,C_3,C_2 \rightarrow 1$ |
| $N_{3,5}$ | $C_3 \rightarrow 5$; $C_2,C_3 \rightarrow 2$; $C_4,C_3 \rightarrow 1$; $C_1,C_2,C_3 \rightarrow 1$; $C_1,C_4,C_3 \rightarrow 1$; $C_3,C_4,C_3 \rightarrow 1$, $C_1,C_2,C_4,C_3 \rightarrow 2$; $C_2,C_3,C_4,C_3 \rightarrow 2$; $C_2,C_1,C_4,C_3 \rightarrow 1$, $C_1,C_2,C_1,C_4,C_3 \rightarrow 1$ |
| $N_{4,5}$ | Node 4 is does not exist |
| $N_{4,5}$ | $C_5 \rightarrow 2$; $C_3,C_5 \rightarrow 1$; $C_4,C_5 \rightarrow 1$; $C_1,C_4,C_5 \rightarrow 1$; $C_2,C_3,C_5 \rightarrow 1$; $C_3,C_4,C_5 \rightarrow 1$; $C_1,C_2,C_4,C_5 \rightarrow 1$; $C_2,C_1,C_4,C_5 \rightarrow 1$; $C_2,C_3,C_4,C_5 \rightarrow 1$; $C_1,C_2,C_1,C_4,C_5 \rightarrow 1$ |

Table 4.3

## 5. Process Flow

Algorithm Procedure that has been discussed in above section can be summarized from the Flow diagram as shown in the Figure 5.1

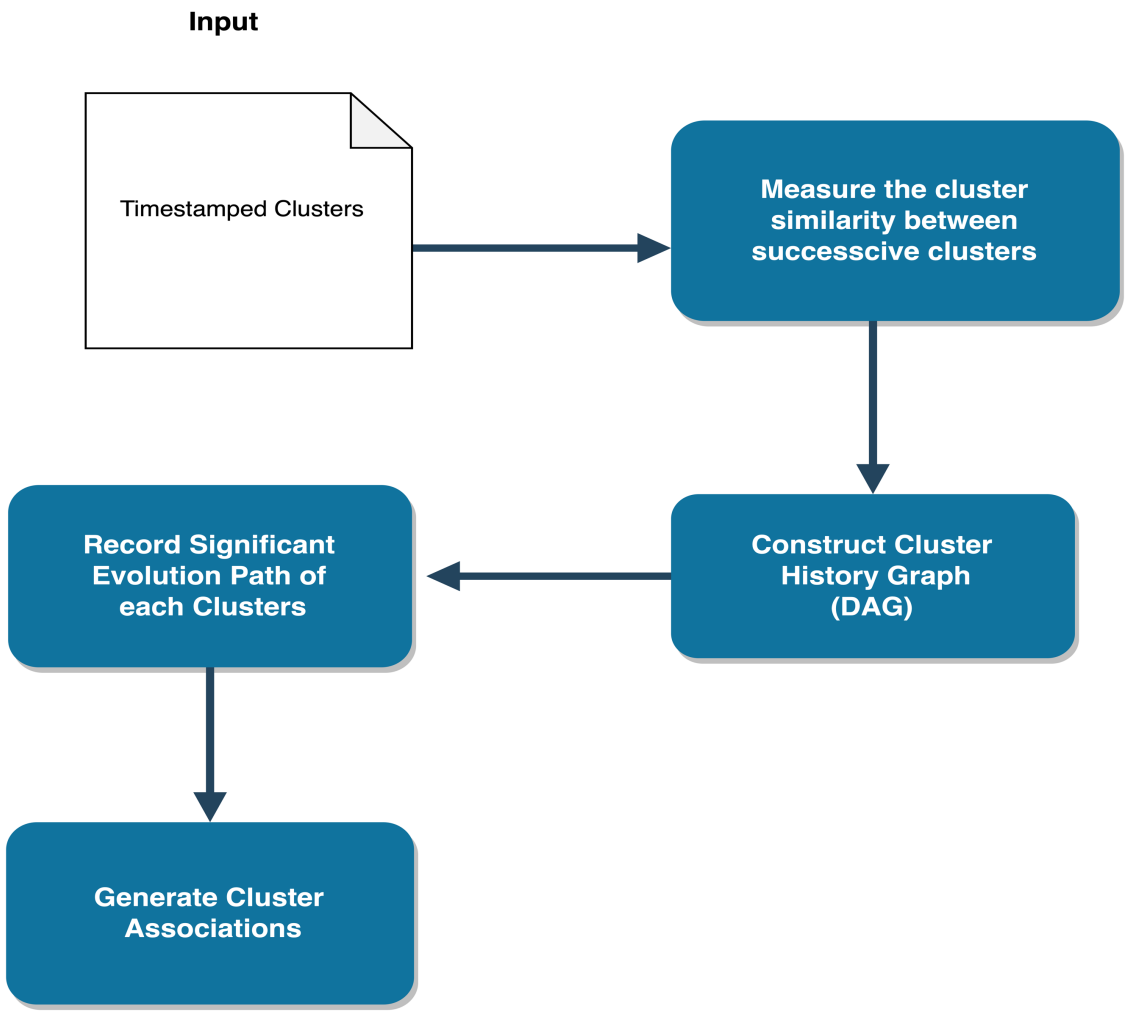


Figure 5.1

## 6. Algorithm

The overall approach of the algorithm is to take the set of timestamped clusters to input process. In section 7 pseudo code of the algorithm is written. In this section describe the major steps of the algorithm.

- First read each time stamped cluster data, and store cluster sub history in a Hash Table Mapping and sort list of mapping in the increasing order of time.
- From the cluster history table (Hash Table) that contains details of cluster and their members at particular time point create Cluster lineage graph.

- For each consecutive pairs of clusters at time $t_i$ and $t_{i+1}$ the temporal relationships between them is calculated. Metric for the temporal relationships between them can be chosen as the distance metric or similarity coefficient. For each pair of consecutive Cluster $C_{i,t}$ and $C_{i,t+1}$ node and weighted edges are added to the CLG.
- A Cluster Lineage Graph (CLG) see figure 4.3 is then created that enables the overall history of the population to be analysed.
- A path update is maintained for each node of the graph associated with its parent node which is achieved from it indegree edges and weights associated.
- Each node at graph generates its ancestor history such that
    - strength of evolution path in terms of coefficient(metric) as an average value of the path traversed so far.
    - Count of same path occurring to particular time and node
    - Frequently associated node during its history

## 7. Pseudocode

**Input** Timestamped cluster generated by clustering algorithm
**Output** Cluster Evolution trajectory path

1: **Read** and Store cluster history of interest between time interval $t_s$ and $t_e$
2: **Select** as input list of clusters at $t_s$ as parent-list
3: **Create** CLG,
    - initialize with entry of nodes from clusters(parent-list) at $t_s$
4: **while** not at interval end time $t_e$
5: **Input** list of clusters at next time point $t_i$+1 as child list
6: **Add** corresponding cluster nodes to CLG child list
7: **foreach** child-cluster in child list
8: **foreach** parent-cluster in parent-list
9: **calculate** similarity coefficient between child-cluster and parent-cluster
10: **if** coefficient greater than **Threshold**

11: **add** edge to CLG
12: parent-node is corresponding parent-cluster
14: child-node is corresponding child-cluster
15: weight is equal to coefficient
16: **Update** cluster lineage path of present child-node
17: **end if**
18: **end foreach**
19: **end foreach**
20: Set child-list as parent-list
21: **end while**
22: **Update** present node
23: **foreach** path from list of parent nodes
24: copy path and add present node to the path
25: update weight of path as geometric mean between edge weight and average coefficient of path
26: update count of path if multiple path exists
27: **end Update**

## 8. Implementation

This section describes the implementation of the algorithm so far discussed. Code base for the algorithm is implemented using JAVA programming language, and in the discussion below may contain java language terminologies however we intend to generalise the discussion in common paradigm suitable for any language.

- Object Mapping using hash table

  Total data population may be described as $X_N = \{x_1, x_2, x_3, x_4, x_5, \ldots\ldots\ldots\ldots, x_N\}$ for given time interval. And any cluster at time t is defined as $C_i(x_t) = \{x_t \in X_N\}$ such that object from the population is the member of particular cluster at particular time. The mapping function is defined such that $h(x_i) \rightarrow x_i$ can map each data point to its corresponding item object. Similarly, encoding function is applied to the individual cluster for mapping such that $h(C_i) \rightarrow C_i$ value can map to its corresponding cluster.

In our implementation hash code of each object is generated which serves as the ID of the object to map its corresponding original object to its Hash Table, which is used for the object storage and retrieval.

For the individual object this hash code is generated from the object label.

Hashcode_of_Object = HashFunction(Item_Label)

In case of Cluster hash code is the function of cluster label and the time point, so hash code is generated from cluster label and the time point.

Hashcode_of_Cluster = HashFunction(Cluster_Label,Time_Point)

Cluster identity problem can be solved using hashcode and mapping object of cluster and its corresponding node. Also, using hashcode of data point helps us treat all the data objects as the data point irrespective of their type like data point represented by multidimensional signature can be identified by the hashcode.

- Similarity measurement

  Jaccard Similarity Function is implemented in following manner.

  Intersection = Find_intersection_between(Set1,Set2)

  Union = Find_Union_between(Set1,Set2)

  Cardinality = calculate_cardinaity(Set)

$$\text{Jaccard\_Similarity}\ (\text{Set1}, \text{Set2}) = \frac{\text{cardinality}\ (\text{Intersection})}{\text{cardinality}\ (\text{Union})}$$

Cosine similarity function is implemented between the two vectors in following manner. If vector1 is represented as $\{x_1,x_2\}$ and vector1 is represented as $\{x_3,x_4\}$

Cosine similarity is implemented as.

Dot_Product = scalar_multiply(vector1, vector2)

Length_vector(vector1) = square_root(Dot_product(vector1,vector1))

$$\text{Cosine similarity}(\text{vector1}, \text{vector2}) = \frac{\text{Dot\_product}(\text{vector1}, \text{vector2})}{(\text{length\_vector}(\text{vector1}) * (\text{length\_vector}(\text{vecto2})}$$

- Construct Cluster Lineage Graph

  For the graph construction initialize the graph with root node. Then iterate the following steps for the given interval of time from start point to end point of time interval as $t_s$ to $t_e$ respectively.

  As explained earlier Node is created with the cluster label and the time signature. Mapping function can retrieve the cluster from hash table for the step of computing similarity coefficient. Also, note that for the ancestor list we use the cluster label to keep record of the sequence path.

  Initialize graph with root node.
  Then add all nodes $N_x$ to root from cluster list at time $t_s$.
  Repeat for $t_s$ to $t_e$
  For each cluster $C_x$ in list of clusters at time t
    - Compute similarity coefficient with all the cluster that exists at time t+1
    - Add edge and Node $N_{i,t+1}$ to graph if similarity coefficient is greater than predetermined threshold value.
    - Add sequence path and its support count to the list of ancestor record list including itself to sequence path.

- Generate graph statistics

  Various statistics can be generated from the cluster lineage graph. Each node keeps the record of its sequence path and their corresponding count which can be generated for the further analysis of the results and visualization purpose. This can be seen in an example from the Table 5.2 and Table 5.3. Furthermore, each edge of the graph has the similarity coefficient or in general metric value. In the path progression the average coefficient of each path can be taken as the geometric mean of its parent path and present coefficient.

  At given time point t we get list of Nodes at that time window.
  For each node $N_{i,t}$ of the list Nodes at t
    - Add ancestor record list to the bag of collection.

Here, bag of collection can be of any data structure which is able collect the list of records. In JAVA this can be the subclass of collection framework like List, LinkedList or Map.

- Finding associative clusters in terms of cluster evolution path trajectories
  The graph at any time point of interest can be analysed from these statistics of their ancestor paths. Also, other important characteristics of the cluster activity can be analysed from the factor like indegree edges and the outdegree edges, average weights of these edges.

## 9. Discussion

It is computationally expensive to repeatedly rescan the entire dataset, however in our approach statistics about the cluster data is generated at first graph building step and information is stored on the corresponding node. At a glance, it may seem this is a memory inefficient approach however this solution is better because usually number of clusters are far small than the actual raw data, so memory is hardly an issue. Input to the algorithm as cluster may contain data point of various nature, like datapoints described by their classes, or multidimensional numerical values or their label. Input cluster are converted or treated by design to be described by the set of data points it contains so it is restricted to a specific cluster definition making this approach equally applicable with partitions, hierarchical clusters or density-based clusters.

Complexity of this algorithm largely depends upon the number clusters and the time span or number of time interval under consideration, however complexity of algorithm in general is $O(n^{2}*m)$ where n is the average number of clusters and m is the number of time interval. In usual case the n is comparatively small, and number of time interval is larger. It is also important to note that complexity does not depend upon the total number of itemset rather it is dependents upon the number clusters. The initial test was conducted using synthetic data where sales record of market basket data is used. Data was basically fabricated to be clustered according to their sales based on their popularity in the fortnightly basis for a year. The test variable was changed on the number of clusters equals to 5 and other with number 10 over the total interval points of 26. As predicted algorithm can successfully record the cluster evolution paths and

indicate that for large amount migrating objects gives the temporal records of paths count higher.

Cluster similarity is used to measure the temporal relationship of any two clusters and as discussed Jaccard similarity index is used for this purpose. As the Jaccard similarity metric dictates the nature of evolution and their evolution characteristics in terms of object migration however Jaccard index fails to characterise the volume of the objects flow from one group to another group. The major bottle neck to record and characterise according to the volume of object flow comes from the algorithm model itself. As algorithm is modelled to accumulate the cluster data as a set thus pairwise distance measure technique is used. Also, its noted that cosine similarity and Jaccard similarity metric yield almost same values while measuring two sets. However, when the object migration and volume of migration are of the interest, we can change the mode of measurement technique from set based to group of objects where frequency of objects are available then this cosine measurement metric can be used for cluster evolution characterisation. Discussion over frequent pattern mining and substructure mining can be further elaborated for the future work. We were able to summarise the history of cluster history trajectory and find the frequent temporal path from graph, able to find the maximal path, find closed frequent path, nevertheless, further investigation is required for association rule mining which can be implemented using any popular mining methods. This work also is basis for further mining for pattern discovery, the paper (Yan and Han,2002) has been designed for frequent pattern discovery by the construction of the frequent pattern tree in a graph data.

Another issue for discussion is how to set the optimal threshold value for temporal characteristics. As there is always a trade-off between use of distance metric and its semantic but the size of evolving clusters, objects migrating pattern, number of clusters will also largely affect the metric values. So, prior knowledge about the data domain and careful examination of metric behaviour being used is required to set the threshold value to generate the significant results.

## 10. Conclusion

At all stage of time point the algorithm is able generate the cluster history and the evolution statistics in terms of path recorded, path count and their average of metric value calculated. As a result, we were able to find the frequent paths and their count. These path trajectories are assigned the metric weightage measured as average of cluster interactions. The purposed model is able to successfully represent the cluster history in to the graph structure, nevertheless, model proves to be the primary tool for the cluster history characterisation. Pairwise method for cluster comparing and tracking progression can significantly characterize the frequent object transfers which is reflected with the help of Jaccard similarity that is applied to measure the object migration to give significant characteristics as a metric. Cluster association can be produced from the frequent substructure in graph furthermore, ancestors of the cluster can be traced back who has significant contribution.

Statistical results are annotated by the path records which are significant factors as cluster evolution summary of individual cluster evolution or evolution of total population as a whole. Algorithm is designed to incorporate cluster and datapoints regardless of data domain and data types moreover, algorithm does not depend on the previous clustering algorithm used to generate clusters.

Regardless of the size of the actual data, algorithm is able to shrink the size of dataset with significant characteristics providing basis for efficient mining and interpretability. Algorithm provides an efficient model of framework for analysing cluster evolution providing result that can be further used for various of applications. Complexity of algorithm and initial performance is acceptable for reasonable volume of data sets however, there are plenty of opportunities for further enhancements.

## Abbreviations

CLG – Cluster Lineage Graph
DAG – Directed Acyclic Graph
DFS – Depth First Search
FP tree – Frequent Pattern Tree
FT – Frequent Trajectory
GPS – Global Positioning System
GSM – Global system for Mobile Communication
HOM – Higher Order Mining
LG – Lineage Graph
VI – Variation of Information

## References:


Aggarwal, C.C., 2005. On change diagnosis in evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, *17*(5), pp.587-600.

Aggarwal, C.C., Han, J., Wang, J. and Yu, P.S., 2003, September. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29* (pp. 81-92). VLDB Endowment.

Agrawal, R. and Srikant, R., 1994, September. Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

Arabie, P. and Boorman, S.A., 1973. Multidimensional scaling of measures of distance between partitions. *Journal of Mathematical Psychology*, *10*(2), pp.148-203.

Borgelt, C. and Berthold, M.R., 2002. Mining molecular fragments: Finding relevant substructures of molecules. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* (pp. 51-58). IEEE.

Chen, C.W.K. and Yun, D.Y., 1998, September. Unifying graph-matching problem with a practical solution. In *Proceedings of International Conference on Systems, Signals, Control, Computers* (Vol. 55).

Cook, D.J. and Holder, L.B. eds., 2006. *Mining graph data*. John Wiley & Sons.
Dehaspe, L., Toivonen, H. and King, R.D., 1998, August. Finding Frequent Substructures in Chemical Compounds. In *KDD* (Vol. 98, p. 1998).

Denœud, L. and Guénoche, A., 2006. Comparison of distance indices between partitions. In *Data Science and Classification*(pp. 21-28). Springer, Berlin, Heidelberg.

Dupplaw, D. and Lewis, P.H., 1999, December. Content-based image retrieval with scale-space object trees. In *Storage and Retrieval for Media Databases 2000* (Vol. 3972, pp. 253-262). International Society for Optics and Photonics.

Fleder, D. and Padmanabhan, B., 2006, December. Cluster evolution and interpretation via penalties. In *Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06)* (pp. 606-614). IEEE.

Gonzalez, J., Holder, L.B. and Cook, D.J., 2001, September. Application of graph-based concept learning to the predictive toxicology domain. In *Proceedings of the Predictive Toxicology Challenge Workshop*.

Hopcroft, J., Khan, O., Kulis, B. and Selman, B., 2004. Tracking evolving communities in large linked networks. *Proceedings of the National Academy of Sciences*, *101*(suppl 1), pp.5249-5253.

Huan, J., Wang, W., Bandyopadhyay, D., Snoeyink, J., Prins, J. and Tropsha, A., 2004, March. Mining protein family specific residue packing patterns from protein structure graphs. In *Proceedings of the eighth annual international conference on Resaerch in computational molecular biology* (pp. 308-315). ACM.

Hubert, L. and Arabie, P., 1985. Comparing partitions. *Journal of classification*, *2*(1), pp.193-218.
Hubert, L., 1977. Nominal scale response agreement as a generalized correlation. *British Journal of Mathematical and Statistical Psychology*, *30*(1), pp.98-103.

Inokuchi, A., Washio, T. and Motoda, H., 2000, September. An apriori-based algorithm for mining frequent substructures from graph data. In *European conference on principles of data mining and knowledge discovery* (pp. 13-23). Springer, Berlin, Heidelberg.

Kalnis, P., Mamoulis, N. and Bakiras, S., 2005, August. On discovering moving clusters in spatio-temporal data. In *International Symposium on Spatial and Temporal Databases*(pp. 364-381). Springer, Berlin, Heidelberg.

Kalviainen, H. and Oja, E., 1990. Comparisons of attributed graph matching algorithms for computer vision. In *In Proc. of STEP-90, Finnish Artificial Intelligence Symposium*.

Kuramochi, M. & Karypis, G., 2004. GREW - a scalable frequent subgraph discovery algorithm. Fourth IEEE International Conference on Data Mining (ICDM'04), pp.439–442.

Meilă, M., 2007. Comparing clusterings—an information based distance. *Journal of multivariate analysis*, *98*(5), pp.873-895.

Mohlin, E., 2015. A metric for measurable partitions. *Journal of Mathematical Psychology*, *67*, pp.39-44.

Müller, H. and Hamm, U., 2014. Stability of market segmentation with cluster analysis–A methodological approach. *Food Quality and Preference*, *34*, pp.70-78.

Ntoutsi, I., Spiliopoulou, M. and Theodoridis, Y., 2011, June. Summarizing cluster evolution in dynamic environments. In *International Conference on Computational Science and Its Applications* (pp. 562-577). Springer, Berlin, Heidelberg.

Rahman, M., Hassan, M.R. and Buyya, R., 2010. Jaccard index based availability prediction in enterprise grids. *Procedia Computer Science*, *1*(1), pp.2707-2716.

Ramon-Gonen, R. and Gelbard, R., 2017. Cluster evolution analysis: Identification and detection of similar clusters and migration patterns. *Expert Systems with Applications*, *83*, pp.363-378.

Rand, W. (1971). Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association, 66(336), pp.846-850.

Roddick, J. & Spiliopoulou, M., 1999. A bibliography of temporal, spatial and spatio-temporal data mining research. ACM SIGKDD Explorations Newsletter, 1(1), pp.34–38.

Roddick, J. et al., 2008. Higher order mining. ACM SIGKDD Explorations Newsletter, 10(1), pp.5–17.
Salton, G., 1989. Automatic text processing: The transformation, analysis, and retrieval of. *Reading: Addison-Wesley*, *169*.

Shaw, A.A. and Gopalan, N.P., 2014. Finding frequent trajectories by clustering and sequential pattern mining. *Journal of Traffic and Transportation Engineering (English Edition)*, *1*(6), pp.393-403.

Spiliopoulou, M., Ntoutsi, E., Theodoridis, Y. and Schult, R., 2013, September. MONIC and followups on modeling and monitoring cluster transitions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*(pp. 622-626). Springer, Berlin, Heidelberg.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y. and Schult, R., 2006, August. Monic: modeling and monitoring cluster transitions. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 706-711). ACM.

Wang, L., Meng, J., Xu, P. and Peng, K., 2018. Mining temporal association rules with frequent itemsets tree. *Applied Soft Computing*, *62*, pp.817-829.
Yan, X. and Han, J., 2002. gspan: Graph-based substructure pattern mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* (pp. 721-724). IEEE.

Zhang, S. and Yang, J., 2008, July. Ram: Randomized approximate graph mining. In *International Conference on Scientific and Statistical Database Management* (pp. 187-203). Springer, Berlin, Heidelberg.

Zhou, A., Cao, F., Qian, W. and Jin, C., 2008. Tracking clusters in evolving data streams over sliding windows. *Knowledge and Information Systems*, *15*(2), pp.181-214.