

A Visual Mapping Approach for Trend Identification in Multi-Attribute Data

Jesse Bockstedt and Gediminas Adomavicius
Department of Information and Decision Sciences
Carlson School of Management, University of Minnesota
(jbockstedt@csom.umn.edu, gedas@umn.edu)

Abstract: Organizations and firms are increasingly capturing more data about their customers, suppliers, competitors, and business environment. Most of this data is multi-attribute (multi-dimensional) and temporal in nature. Data mining and business intelligence techniques are typically used to discover patterns in this data; however, mining meaningful temporal relationships is often difficult. We introduce a new temporal data analysis and visualization technique for representing trends in multi-attribute temporal data using a clustering-based approach. We define a new analytical construct called the temporal cluster graph which maps multi-attribute temporal data into a two-dimensional trend graph that clearly identifies trends in dominant data types over time. We also present C-TREND, a system that implements the proposed technique, and demonstrate applications of technique by analyzing the change in technical characteristics of wireless networking technologies over a six year period.

1. Introduction

Organizations and firms are increasingly capturing more transactional data, containing multiple attributes and some measure of time, e.g., through their websites, e-commerce firms capture clickstream and purchasing behavior of their customers. One of the common analysis tasks for firms is to determine whether trends exist in their transactional data; e.g., a retailer may wish to know if the types of its regular customers are changing over time or a financial institution may wish to determine if the major types of credit card fraud transactions change over time. Visualizing and analyzing this type of data can be extremely difficult because it can have numerous attributes (dimensions). Additionally, it is often desired to aggregate over the temporal dimension (by day, month, quarter, year, etc.) to match corporate reporting standards. The approach that we take is to mine the data according to specific time periods and then compare the data mining results across time periods to discover similarities.

Consider the plot of a retailer's customers by age and income over three months in Fig. 1. X's represent customers in the 1st month, triangles represent customers in the 2nd month, and circles represent customers in the 3rd month. An analyst may be tasked with the job of discovering trends in customer type over these three months. In the initial representation, patterns in the data and relationships over time are difficult to identify. However, partitioning the data by time leads to the identification of clusters within each period. Clusters encapsulate similar data points and identify common types of customers. The final representation in Fig. 1 is a mapping of the multi-dimensional temporal data into an intuitive analytical construct that we call a *temporal cluster graph*. These graphs contain important information about the relative proportion of common transaction types within each time period, relationships and similarities between common transaction types across time periods, and trends in common transaction types over time. In this paper, we use a design science approach [6] to develop this new analytical construct and discuss its implementation, the C-TREND system (Cluster-based Temporal Representation of Event Data), which uses this construct to provide a novel approach to visualization and analysis of multi-attribute transactional data.

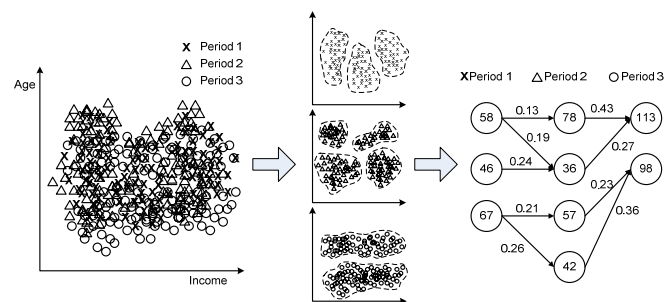


Fig. 1. Reducing multi-attribute temporal complexity by partitioning data into time periods and producing a temporal cluster graph.

2. Conceptual Foundations

Temporal data mining (TDM) is a growing stream in the knowledge discovery research field, and the technique we propose relates well to this stream. The goal of TDM is to discover relationships between events and sequences of events that have some form of temporal dependency [1]. TDM approaches depend on the nature of the event sequence being studied. For example, time series analysis [2, 9, 11] is used to mine a sequence of continuous real-valued elements, and is often regression-based, relying on the definition of a model. Moreover, time series analysis techniques typically are examples of supervised learning; in other words, they estimate the effects of a set of independent variables on a dependent variable. Another common TDM technique is sequence analysis [10, 15]. Sequence analysis is often used when the sequence is composed of a series of nominal symbols [1]; examples of sequences include genetic codes and the click patterns of website users. Sequence analysis is designed to look for the recurrence of patterns of specific events and typically does not account for events described with multi-attribute data [1].

In the business intelligence context, trend discovery is often better addressed using unsupervised learning techniques because models of trends and specific relationships between variables are not typically known. The technique proposed in this paper follows this approach and, therefore, provides an advantage over time series analysis. Specifically, the proposed technique uses clustering, i.e., the unsupervised discovery of groups in a data set [7], as the basis for analyzing trends. Basic clustering strategies can be separated into hierarchical and partitional, and all use some form of a distance or similarity measure to determine cluster membership and boundaries [7]. Our technique uses clusters identified in multiple time periods and identifies trends based on similarities between clusters over time. This is essentially a temporal clustering approach that builds on existing clustering methods and complements existing TDM approaches.

Data mining also requires the inclusion of the human in the data exploration process in order to be effective [8]. *Visual data exploration* is the process of presenting data in some visual form and allowing the human to interact with the data to create insightful representations [8]. It typically follows the information-seeking mantra [12]: *overview, zoom and filter, details-on-demand*. With the dramatic increase in the amount of data being captured by organizations, multidimensional visualization techniques have become an important area of data mining research. Representing multidimensional data in a 2- or 3-dimensional visual graphic cannot be achieved through simple mapping, and many data visualization techniques have been developed (for overviews, see [3, 4, 13, 14]). We recognize the importance of visualization in TDM and in this paper present a new technique designed to aid in multi-attribute temporal data visualization.

3. Constructs and Overview of the Technique

We define a new analytical construct called the *temporal cluster graph*. This graph is obtained as a result of several steps. First, a transactional data set D is partitioned based on time periods into t data subsets D_1, \dots, D_t . Data within each partition is then clustered using one of the standard clustering techniques, such as k -means or hierarchical approaches [5], and k_i represents the number of clusters obtained for the i^{th} data partition. The directed graph G consists of a set of nodes V and a set of directed edges E , i.e., $G = (V, E)$. The total set of graph nodes consists of t subsets of nodes $V = \{V_1, V_2, \dots, V_t\}$, where each subset corresponds to a data partition and contains k_i nodes. The node $v_{i,j} \in \{V_i\}$ is the j^{th} node in the i^{th} partition. Nodes are labeled with the size of the cluster they represent (i.e., the number of data points in the cluster). Edges $e(v_{i,j}, v_{i+1,k}) \in E$ connect nodes in adjacent partitions and are labeled with a distance value between the two nodes (i.e., the similarity between the clusters represented by the nodes incident to the edge). A variety of distance metrics (e.g., Euclidean, Chebyshev) and cluster distance measures, such as max-link and average-link [5], could be used to determine this value. The temporal cluster graph is a general-purpose construct and does not depend on specific choices of the clustering method and distance metrics.

Edges in a temporal cluster graph are possible between nodes only in different partitions, in other words, $e(v_{i,j}, v_{i,k}) \notin E$. For our initial development, we limit this to adjacent partitions to reduce

visualization complexity. It is possible to calculate edges across non-adjacent partitions, and this is an extension we hope to address in future research. To identify temporal trends, edges are directed only from earlier partitions to later partitions, in other words, $e(v_{i,j}, v_{i-1,k}) \notin E$ and partitions are indexed chronologically for $i=1, 2, \dots, t$. The representation of data on the right side of Fig. 1 provides an example of a temporal cluster graph that contains three data partitions.

Following the information seeking mantra of data visualization [12], we define two parameters that can be used to filter and adjust the temporal cluster graph elements. It is possible that not all clusters will be large enough to be considered relevant to the analysis at hand. For example, in a data set of 2000 data points, a cluster of size $s = 2$ (i.e., containing only two data points) would likely be spurious for many practical applications. To address this issue and provide more data visualization control to the user of the temporal cluster graph, we introduce a parameter that can be used to determine if nodes generated by the initial clustering solution are “strong” enough to be included in the trend analysis. For every data partition D_i , the clustering solution contains k_i clusters, and some of these clusters can be filtered out based on the global *within-period trend strength* parameter $\alpha \in [0, 1]$. In other words, each data partition utilizes the same value of α . Let V_i be the set of clusters in data partition D_i , i.e., V_i contains the k_i clusters identified in the clustering solution for D_i . For every cluster $j \in V_i$, if the cluster size $s_j \geq \alpha |D_i|$, then cluster j is included as a node in the output graph. $\alpha |D_i|$ is the minimum node size threshold for data partition D_i , where $|D_i|$ is the number of data points in D_i . As an example, consider a partition with 200 data points; changing $\alpha = 0$ to

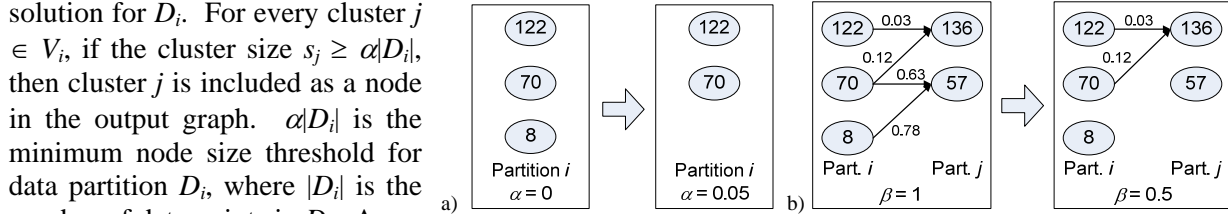


Fig. 2. (a) changing the within-period trend strength, (b) changing the cross-period trend strength

$\alpha = 0.05$ would filter out any clusters with less than ten data points, or 5% of the total data points in the partition (see Fig. 2a).

Since an edge is possible between any two nodes in adjacent partitions, it is desirable to limit the edges included in a graph to those that are incident to “very similar” nodes, thus representing a continuous trend over time. For this purpose, we introduce the *cross-period trend strength* parameter β that is used to filter out spurious edges based on their weight. An edge is included in the output graph if it meets two criteria: (a) the edge is incident to two nodes that are included in the output graph (as determined by the clustering solution and within-period trend strength α); and (b) the edge weight is less than or equal to a threshold η which depends on cross-period trend strength β . The edge threshold η is calculated by taking the average of weights for all the possible edges among the nodes in two adjacent data partitions (say i and $i+1$) and adjusting it by the user-specified β parameter:

$$\eta_{i,i+1} = \beta \times [\text{average edge length between partitions } i \text{ and } i+1] = \beta \left(\frac{\sum_{m=1}^{k_i} \sum_{n=1}^{k_{i+1}} d(v_{i,m}, v_{i+1,n})}{k_i k_{i+1}} \right).$$

Here $v_{i,m}$ is the m^{th} node in the i^{th} partition and $d(v_{i,m}, v_{i+1,n})$ is the distance between nodes $v_{i,m}$ and $v_{i+1,n}$. From the definition of β and the relationship between β and η , it is easy to see that lowering β will result in more edges being filtered out, leaving only the strongest trends displayed in the graph. $\beta \in [0, 1]$ and, similarly to α , β is a global parameter value for the entire data set. The intuition for η is that edges that are weaker than the “average” overall edge are likely to be uninteresting and, therefore, analysis should start with the average weighted edge.

This procedure can accommodate a variety of distance metrics (e.g., Manhattan, Chebyshev) and cluster similarity measures, such as average- and min-link [5]. Note that if $d(x,y) = 0$ then clusters x and y are identical with respect to the chosen metrics, and therefore most likely have a very similar makeup. As an example, consider two partitions, i and j , where the average edge weight between these partitions is 1.21. Changing $\beta = 1.0$ to $\beta = 0.5$ will filter out any edges with weights greater than $\eta = 0.605$ (see Fig. 2b).

We demonstrate the use of temporal cluster graphs by analyzing a real-world transactional attribute-value dataset of over 3000 certifications for new wireless networking technologies, based on the IEEE 802.11 standard, awarded by the Wi-Fi Alliance (wi-fi.org). Wi-Fi Alliance certifications are awarded for ten different technology categories: access points, cellular convergence products, compact flash adapters, embedded clients, Ethernet client devices, external cards, internal cards, PDAs, USB client devices, and wireless printers. Each product certification consists of a date of certification and a set of binary attributes indicating the presence or absence of specific technical capabilities (e.g., compliance with 802.11g).

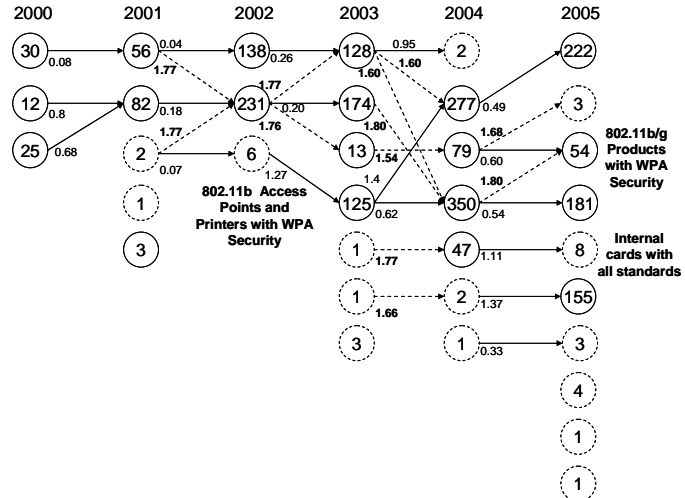


Fig.3. Trends in Wi-Fi alliance yearly certifications with $\alpha = 0.00, \beta = 1.0$ (solid and dashed lines) and $\alpha = 0.02, \beta = 0.8$ (solid lines only).

Edges were rendered between nodes in adjacent time periods to represent similarities between clusters over time; e.g., the edge labeled 0.08 in Fig. 3 indicates that the center of the cluster labeled “30” in 2000 is at a distance of 0.08 from the center of the cluster labeled “56” in 2001. This suggests the clusters are extremely similar. Following this same trend into 2002, we see an edge with weight 0.04 and, more generally, that during 2000-2002 802.11b access points with very similar technical specifications constituted one dominant technology type in the set of all available Wi-Fi technologies. After 2002, we see a larger deviation with an edge weight of 0.26 and then the trend ends indicating a lack of similarity between the node labeled “128” in 2003 and any of the nodes in 2004. There is a node of access points in 2004, labeled “277”; however, the average technical characteristics of the technologies making up this node are significantly different than the previous generation. Specifically, the introduction of 802.11g and security technologies led to a significantly different class of wireless access points.

The temporal cluster graph in Fig. 3 uses two sets of graph parameters. The solid edges and nodes represent filtering using $\alpha = 0.02$ and $\beta = 0.8$, where solid plus dashed elements represent more relaxed parameters with $\alpha = 0.0$ and $\beta = 1.0$. The advantage of adjusting α and β is that it provides the C-TREND user the ability to show or hide possible trends according to their strength. For example, the more relaxed parameters allow the C-TREND users to uncover a new cluster in 2002 and identify a trend of 802.11b access points and printers with WPA security that was not apparent with the more strict parameters. On the other hand, there are many clusters of size 1 with no adjoining edges. These are likely to be isolated events that do not provide insights on trends and can be filtered out of the graph using more restrictive parameters.

4. Implementation and Performance

C-TREND is the system implementation of the temporal cluster graph identification and visualization technique; it provides an end-user with the ability to generate graphs from data and adjust the graph parameters. C-TREND consists of two components: a preprocessing component written in C and a GUI for interactive data visualization written in Java. In the preprocessing phase, the data set is partitioned based on time periods, and each partition is clustered using one of many traditional clustering techniques, such as k -means or hierarchical approaches [5]. The results of the clustering for each partition are used to generate two data structures: the node list and the edge list. Creating these lists in the preprocessing phase allows for more effective (real-time) visualization updates of the C-TREND output graphs. In the interactive analysis phase, C-TREND allows the user to modify the α and β parameters on demand to produce real-time filtering and aggregation of the data.

The computationally costliest preprocessing operation is the generation of the edge list, which is essentially a list of ordered pairs, each pair representing adjacent nodes that define an edge. Due to the limitations of human cognitive abilities, in our implementation we assume that each partition will

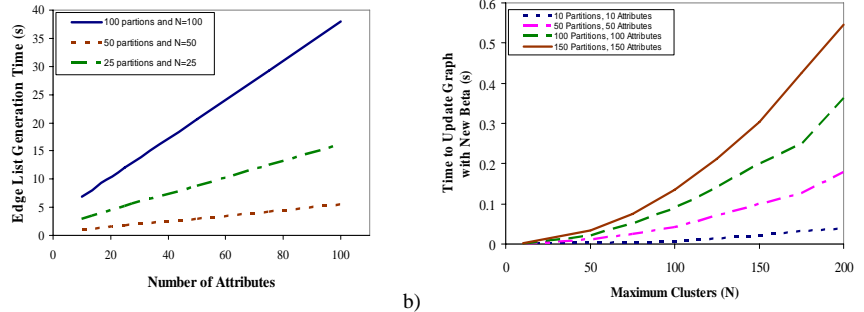


Fig. 4. Computational performance experiments on: (a) edge list creation, and (b) changing cross-period trend strength parameter β .

contain up to N nodes (i.e., N is the maximum-sized clustering solution for a partition). Therefore, there can be a maximum of N^2 possible edges between two adjacent data partitions and $(t-1)*N^2$ possible edges for the entire data set, where t is number of partitions, with time complexity asymptotic upper bound of $O(N^2t)$. By pre-calculating all possible edges and their weights in the preprocessing phase, C-TREND can achieve real-time functionality in the analysis phase, as the output graph parameters are being interactively adjusted. Fig. 4a shows that, even in some extreme visualization cases where the number of clusters in each partition is $N=100$ or 500 , edge list preprocessing takes less than one minute.

Theoretical complexity of the parameter-changing operations can be computed based on the maximal number of clusters N and the number of data partitions t . C-TREND checks the status of each edge and node to determine if it should be rendered. If each partition contains N possible nodes and there are t partitions, changing α has an asymptotic upper bound of $O(Nt)$. Since there are $(t-1)N^2$ possible edges, changing β is $O(N^2t)$. In Fig. 4b, we empirically show that, even as N increases, C-TREND can still update β relatively quickly.

5. Discussion

To provide additional analytical power to the users, we have begun to develop metrics to measure different aspects of the paths identified in the temporal cluster graphs. An example of one metric is the *trend directionality ratio*, $dir(v_0, v_n)$, the ratio of the direct distance and the path distance: $dir(v_0, v_n) = d_D(v_0, v_n)/d_P(v_0, v_n)$. Here the *direct distance* $d_D(v_0, v_n)$ is the distance measured directly between nodes v_0 and v_n , and the *path distance* $d_P(v_0, v_n)$ is the total distance along a specific path between nodes v_0 and v_n . Because $0 \leq d_D(v_0, v_n) \leq d_P(v_0, v_n)$, we have $dir(v_0, v_n) \in [0, 1]$. Specifically, the directionality ratio measures the continuity of consistent directional change in a trend. A value of $dir(v_0, v_n)=1$ means that $d_D(v_0, v_n) = d_P(v_0, v_n)$ and, therefore, changes in the cluster centers were consistently *in the same direction* throughout the trend for every attribute. Alternatively, $dir(v_0, v_n)$ that is close to zero indicates that $d_D \ll d_P$. In either case, the trend is not moving consistently in the same direction through multidimensional space. Fig. 5 depicts two scenarios for measuring the directionality ratio on path $P = (c_1, c_2, c_3, c_4)$, where clusters c_1, c_2, c_3, c_4 would appear as nodes in partitions t_1, t_2, t_3, t_4 of a graph, respectively. From the figure it is clear that $dir(c_1, c_4)$ is greater in scenario 1 than in scenario 2, and it therefore follows that in scenario 1 the path of cluster centers is moving more consistently in the same direction over time than in scenario 2.

Temporal cluster graphs and C-TREND can be applied in a wide variety of data analysis settings. For example, in business applications, the C-TREND system could be used to: identify changes in customer purchasing behaviors over time, visualize trends in website usage behavior, or identify patterns of credit card use for fraud detection. Practically any scenario in which an analyst wishes to visualize changes in dominant data types over time could utilize temporal cluster

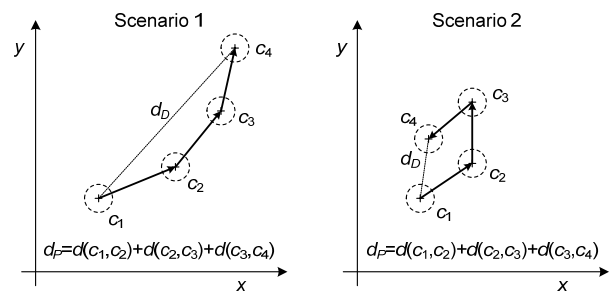


Fig. 5. Two scenarios for calculating the trend directionality ratio.

graphs. In addition, historical modeling of trends in economic and technological change using temporal cluster graphs could aid in the development of forecasts. Possible future extensions of the technique include hypothesis testing and automated data analytics of temporal data using C-TREND as the analytical engine.

Clustering methods do have their limitations. Large numbers of dimensions (attributes) can dramatically increase time complexity of cluster analysis and, since our proposed method relies on clustering, this could impact the technique's performance. Additionally, clustering method effectiveness may be significantly affected by the distance metrics and cluster similarity measures used. For this reason, we developed our analytical constructs independent of specific distance types. Also, the interpretation of the results of a clustering algorithm can be fairly subjective. For this reason, we provide the users the ability to adjust graph parameters and leverage their domain expertise in the construction of temporal cluster graphs.

6. Conclusion

By harnessing computational techniques of data mining, we have developed a temporal clustering technique for discovering, analyzing, and visualizing trends in multi-attribute temporal data. The proposed technique is versatile and gives significant data representation power to the user – domain experts have the ability to adjust parameters and clustering mechanisms to fine-tune trend graphs. It is also scalable: the time required to adjust trend parameters is quite low even for larger data sets, which provides for real-time visualization capabilities. The proposed technique is applicable in many data analysis contexts, and can provide insights for analysts performing historical analyses and generating forecasts.

References

- [1] C.M. Antunes, A.L. Oliveira, "Temporal Data Mining: an overview," In *KDD 2001 Workshop on Data Mining*, San Francisco, August 2001, pp. 1-13.
- [2] P. Brockwell, R. Davis, *Time Series: Theory and Methods*, Springer-Verlag, New York, 2001.
- [3] S. Card, J. Mackinlay, B. Schneiderman, *Readings in Info. Visualization*, Morgan Kaufmann, 1999.
- [4] M.C.F. de Oliveira, H. Levkowitz, "From Visual Data Exploration to Visual Data Mining: A Survey," *IEEE Trans. on Visualization and Computer Graphics*, vol. 9, no. 3, pp. 378-394, 2003.
- [5] R. Duda, P. Hart, D. Stork, *Pattern Classification*, 2nd ed., Wiley-Interscience, New York, 2000.
- [6] A.R. Hevner, S.T. March, J. Park, S. Ram, "Design science in information systems research," *MIS Quarterly*, (28:1), March 2004, pp. 75-105.
- [7] A. Jain, M. Murty, P. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, 31(3):264-323, 1999.
- [8] D.A. Keim, "Information Visualization and Visual Data Mining," *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 100-107, 2002.
- [9] E. Keogh, S. Kasetty, "On the need for time series data mining benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, (7:4), 2003, pp. 349-371.
- [10] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, O. Chen, U. Dayal, M.C. Hsu, "Mining sequential patterns by pattern-growth: The PrefixSpan approach," *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1-17, 2004.
- [11] J. Roddick, M. Spiliopoulou, "A survey of temporal knowledge discovery paradigms and methods," *IEEE Transactions on Knowledge and Data Engineering*, (14:4), 2002, pp.750-767.
- [12] B. Schneiderman, "The eye have it: A task by data type taxonomy for information visualizations," *Proc. IEEE Symp. Visual Languages*, 1996.
- [13] B. Spence, *Information Visualization*, Pearson Education Higher Education Publishers, UK 2000.
- [14] C. Ware, *Information Visualization: Perception for Design*, Morgan Kaufman, 2000.
- [15] M. Zaki, "SPADE: An efficient algorithm for mining frequent sequences," *Machine Learning*, 42(1/2):31-60, 2001.