C-TREND: A New Technique for Identifying and Visualizing Trends in Multi-Attribute Transactional Data

Gedas Adomavicius and Jesse Bocsktedt
Information and Decission Sciences Department
Carlson School of Management, University of Minnesota, Minneapolis
gedas@umn.edu, jbockstedt@csom.umn.edu

Last updated: June 21, 2007

Currently Under Review at IEEE Transactions on Knowledge and Data Engineering (first round)

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 often used to discover patterns in such data; however, mining temporal relationships typically is a complex task. We propose a new data analysis and visualization technique for representing trends in multi-attribute temporal data using a clustering-based approach. We introduce C-TREND, a system that implements the temporal cluster graph construct, which maps multi-attribute temporal data to a two-dimensional graph that clearly identifies trends in dominant data types over time. In this paper, we present our temporal clustering-based technique, discuss its algorithmic implementation and performance, demonstrate applications of the technique by analyzing data on wireless networking technologies and baseball batting statistics, and introduce a set of metrics for further analysis of discovered trends.

Index Terms— Clustering, data and knowledge visualization, data mining, interactive data exploration and discovery.

1 INTRODUCTION

Business intelligence applications represent an important opportunity for data mining techniques to help firms gather and analyze information about their performance, customers, competitors, and business environment. Knowledge representation and data visualization tools constitute one form of business intelligence techniques that present information to users in a manner that supports business decision-making processes. In this paper, we develop a new data analysis and visualization technique that presents complex multi-attribute temporal data in a cohesive graphical manner by building on well-established data mining methods. Business intelligence tools gain their strength by supporting decision makers, and our technique helps the users leverage their domain expertise to generate knowledge visualization diagrams from complex data and further customize them.

Organizations and firms are increasingly capturing more data, and this data is often transactional in nature, containing multiple attributes and some measure of time. For example, through their websites, e-commerce firms capture clickstream and purchasing behavior of their customers, and manufacturing companies capture logistics data (e.g., on the status of orders in production or shipping information). One of the common analysis tasks for firms is to determine whether trends exist in their transactional data. For example, a retailer may wish to know if the types of its regular customers are changing over time, a financial institution may wish to determine if the major types of credit card fraud transactions change over time, and a web site administrator may wish to model changes in website visitors’ behavior 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 (e.g., by day, month, quarter, year, etc.) to match corporate reporting standards. The approach that we take in the paper for addressing these types of issues 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 Figure 1. X’s represent customers in the first month, triangles represent customers in the second month, and circles represent customers in the third 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

Fig. 1. Reducing multi-attribute temporal complexity by partitioning data into time periods and producing a temporal cluster graph.
by time clearly leads to the identification of clusters within each period. Clusters encapsulate similar data points and identify common types of customers. Note, that in this example we used only two dimensions (age and income) for more intuitive visualization. In many real-life applications the number of dimensions could be much higher, which further emphasizes the need for more advanced trend visualization capabilities. The third representation in Figure 1 is a mapping of the multi-dimensional temporal data into an intuitive analytical construct that we call a temporal cluster graph. As will be discussed in more detail in the remainder of this paper, 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 summary, the main contribution of this paper is the development of a novel and useful approach for visualization and analysis of multi-attribute transactional data based on a new temporal cluster graph construct, as well as the implementation of this approach as the C-TREND system (Cluster-based Temporal Representation of EveNt Data). The rest of the paper is organized as follows. Section 2 provides an overview of related work in the temporal data mining and visualization research streams. Section 3 introduces the temporal cluster graph construct and describes the technique for mapping multi-attribute temporal data to these graphs. Section 4 discusses the algorithmic implementation of the proposed technique as the C-TREND system and includes performance analyses. Section 5 presents an evaluation of the technique using real-world data on wireless networking technology certifications. Section 6 is a discussion of possible applications, trend metrics, limitations associated with the proposed technique, and a brief discussion of future work. The conclusions are provided in Section 7.

2 RELATED WORK

The research field of data mining has developed a number of methods for identifying patterns in data to provide insights and decision support for users. Data mining and business intelligence approaches are often used for class identification and data visualization in knowledge management systems (Shaw et al. 2001). Increasingly, knowledge discovery in data (KDD) techniques are providing new analytical structures that complement and sometimes replace existing human-expert-based techniques to provide improved decision making support (Apte et al. 2002). Identifying and visualizing temporal relationships (e.g., trends) in data constitutes an important problem that is relevant in many business, scientific, and academic settings. In this section, we provide a brief review of related research in the temporal data mining and visualization streams.

2.1 TEMPORAL DATA MINING

Temporal data mining is a growing stream in the knowledge discovery research field, and the technique we propose relates well to this stream. The goal of temporal data mining is to discover relationships among events and sequences of events that have some form of temporal dependency (Antunes and Oliviera 2001). Temporal data mining has the capability of mining activity rather than just states, which can lead to inferences about relationships and cause-effect association (Roddick and Spiliopoulou 2002). These techniques provide promise of understanding why rather than merely what (Roddick and Spiliopoulou 2002).

Antunes and Oliviera (2001) note that two fundamental problems that must be addressed in temporal data mining are the representation of data and data preprocessing. In addition, Roddick and Spiliopoulou (2002) distinguish between two tactical goals of temporal data mining: prediction of the values of a population’s characteristics, and the identification of similarities among members of a population. They also categorize the temporal data mining research along three dimensions: datatype, mining paradigm, and temporal ordering.

Temporal data mining approaches depend on the nature of the event sequence being studied. For example, time series analysis (Brockwell and Davis 2001; Keogh and Kasetty 2003; Roddick and Spiliopoulou 2002) 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 temporal data mining technique is sequence analysis (Pei et al. 2004; Zaki 2001). Sequence analysis is often used when the sequence is composed of a series of nominal symbols (Antunes and Oliviera 2001); 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 (Antunes and Oliviera 2001).

Discovering and describing trends in temporal data is an important type of pattern discovery commonly used in studying the evolution of objects within a population (Roddick and Spiliopoulou 2002). In much of this research, similarities between multiple sequences or time series are calculated to determine consistencies in trends over time. Examples of this research can be found in Berndt and Clifford (1995), Keogh and Pazzani (1998), and Agrawal et al. (1995).

In the business intelligence context, trend discovery may be better addressed using unsupervised learning techniques because models of trends and specific relationships between variables may not be known. The technique proposed in this paper follows this approach and uses cluster analysis as a basis for identifying trends. Specifically, clustering is the unsupervised discovery of groups in a data set (Jain et al. 1999). The 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 (Jain et al. 1999). The technique proposed in this pa-
per 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 temporal data mining approaches.

### 2.2 Data Visualization

Data mining requires the inclusion of the human in the data exploration process in order to be effective (Keim 2002). 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 (Keim 2002). It typically follows the information seeking mantra (Schneidermann 1996): overview, zoom and filter, details-on-demand. Most formal models of information visualization are concerned with presentation graphics (Bertin 1983, Mackinlay 1986) and scientific visualization (Beshers and Feiner 1993&1994, Senay and Ignatiou 1994, Roth and Mattis 1990, Hibbard et al. 1994). Keim (1997, 2002) and Keim and Kriegel (1994) provide taxonomies of visualization data exploration approaches and note that these techniques can be classified by (a) the type of data, (b) visualization technique, and (c) interaction techniques.

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 two- or three-dimensional visual graphic cannot be achieved through simple mapping, and many data visualization techniques have been developed (for overviews, see de Oliveira and Levkowitz 2003, Card et al. 1999, Ware 2000, and Spence 2000).

Two visualization approaches relevant to the present research are hierarchical techniques and graph-based techniques. Hierarchical techniques subdivide multidimensional space and present the resulting subspaces in a hierarchical fashion (de Oliveira and Levkowitz 2003, Chen 1999). For well-known examples, see the n-Vision system (Bershers and Feiner 1990, 1994), dimensional stacking (LeBlanc et al. 1990), and treemaps (Schneidermann 1992). Graph-based visualization techniques generate large graphs using layout algorithms and abstraction techniques to convey relational meanings clearly and quickly (de Oliveira and Levkowitz 2003). For examples of applications, see Abello and Korn (2001), Battista et al. (1999), Becker et al. (1997), Eick and Wills (1993), Hendley et al. (1995).

Interaction techniques provide the user the ability to dynamically change visual representations (Keim 2002) and can empower the user’s perception of information (de Oliveira and Levkowitz 2003). A comprehensive framework for user interface techniques used in visualization systems can be found in Chuah and Roth (1996). Interactive filtering involves dynamically partitioning a data set into segments and focusing on interesting subsets by either direct selection or specification of subset properties (Keim 1997, Keim 2002, Fishkin and Stone 1995, Tang et al. 2001). Interactive zooming is a common interaction technique that provides the user with a variable display of data at different levels of analysis (Keim 1997, Keim 2002). Zooming capabilities mean that the data representation can be automatically changed to present more or fewer details on demand. Some examples of applications that use interactive zooming include TableLens (Rao and Card 1994) and PAD++ (Bederson 1994).

### 3 Temporal Cluster Graphs

In this paper we present a new data mining technique for identifying and visualizing trends in multidimensional temporal data. We build on both temporal data mining techniques and visual data exploration techniques, and develop a tool that provides the user with the ability to interact with a temporal cluster graph data visualization. Temporal cluster graphs use hierarchical and graph based techniques to explore temporal data, and provide interactive filtering and zooming capabilities for visualization.

#### 3.1 Temporal Cluster Graph Definition

We define a new analytical construct called the temporal cluster graph. This graph is obtained as a result of several steps. First, transactional data set $D$ is partitioned based on time periods into $t$ data subsets $D_1, ..., D_t$ (indexed chronologically) and each $D_i$ is a multi-attribute subdataset containing records with $m$ number of attributes. Data within each partition is then clustered using one of the standard clustering techniques, such as k-means or hierarchical approaches (Duda et al. 2000), and $k_i$ represents the number of clusters obtained for the $i^{th}$ data partition. The temporal cluster graph $G$ is a directed graph that 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 $V = \{V_1, V_2, ..., V_t\}$, where each subset corresponds to a data partition and contains $k_i$ nodes (i.e., each graph node represents a different cluster). The node $v_{ij} \in V_i$ 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 that cluster). Edges $e(v_{ij}, v_{ik}) \in E$ connect nodes in adjacent partitions and are labeled with a distance value between the two nodes, thus representing the similarity between the clusters (nodes) connected by the edge. A variety of distance metrics (e.g., Euclidean, Chebyshev) and cluster distance measures, such as max-link and average-link (Duda et al. 2000), 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.

To identify truly temporal trends, edges are directed only from earlier partitions to later partitions, in other words, $e(v_{ij}, v_{ik}) \notin E$ where $x \leq i$. The representation of data on the right side of Figure 1 provides an example of a temporal cluster graph that contains three data partitions. The first partition contains three nodes, each representing a data cluster identified in the customer data for time period 1 and labeled with the corresponding cluster size. The second and third partition nodes are determined in the same manner for time periods 2 and 3. The edges connect nodes in adjacent partitions, are directed from earlier time periods to later time periods, and are labeled with distances
between nodes.

3.2 GRAPH PARAMETERS
Since one of the main goals of a visual mapping technique is to reduce complexity and present information in a useful and intuitive manner, it is necessary to provide a means for displaying information at different levels of analysis (i.e., facilitate interactive zooming) as well as filtering spurious entities from a temporal cluster graph (i.e., facilitate interactive filtering). We next define three graph parameters that are designed to meet this need.

PARTITION ZOOM
In C-TREND, we refer to the ability to dynamically change the size of the clustering solution in a data partition as the zoom feature. Specifically, each data partition $D_i$ has a corresponding $k_i$ value, where $k_i$ refers to the number of clusters estimated in the clustering solution for that partition. For example, a value of $k_i = 5$ corresponds to the clustering solution for the $i^{th}$ partition that contains exactly 5 clusters.

Note, however, that performing agglomerative hierarchical clustering on a set of $n$ data points results in $n$ different clustering solutions, one at each level of the agglomeration (i.e., the first level has $n$ clusters and each successive solution contains one less cluster). If hierarchical clustering is used on a data partition of size $n$, the displayed solution is just one of the $n$ possible clustering solutions, and the user may be interested in seeing how the temporal trends look given a different solution (i.e., different level of aggregation). The $k_i$ value here is analogous with $k$ in the case of $k$-means clustering, where the user typically specifies a $k$ value upfront, and the data are then partitioned into $k$ clusters. C-TREND provides the user with the control to adjust and visualize the clustering solution for each time partition in real time. For example, in a partition with 100 data points, changing $k_i = 3$ to $k_i = 2$ would re-cluster the data points from three clusters into two clusters (Figure 2).

In all clustering methods, there is variability in the solution based on the user’s understanding of the data and his interpretation of the output. Many clustering techniques assume that the number of clusters is known ahead of time (e.g., $k$-means clustering) and, therefore, a common problem in cluster analysis is deciding the optimal number of clusters that are present in a data set. The C-TREND zoom feature allows the users to apply their domain expertise by adjusting in real time the underlying clustering solution used to build a trend graph and interactively evaluate multiple trend views.

WITHIN-PERIOD TREND STRENGTH
As discussed in the previous section, nodes of the trend graph are created by clustering each data partition to identify common naturally-occurring patterns in data. However, 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. Additionally, it is possible to have singleton clusters appear in the final cluster solution for some clustering approaches, such as agglomerative hierarchical clustering (Duda et al. 2000). To address these issues and provide more data visualization control to the user of temporal cluster graphs, we introduce a user-specified parameter that can be used to determine if nodes generated by the 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 within-period trend strength parameter $\alpha$.

The $\alpha \in [0, 1]$ parameter is global for the entire data set. In other words, each data partition utilizes the same value of $\alpha$. 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. Thus, $\alpha |D_i|$ 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 $\alpha = 0.05$ would filter out any clusters with fewer than 10 data points, or 5% of the total data points in the partition (see Figure 3).

CROSS-PERIOD TREND STRENGTH
In temporal cluster graphs, edges are used to represent relationships between nodes (clusters) in adjacent time partitions. 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

---

1 This could also be addressed to some extent by using outlier detection methods in the data preprocessing phase (Pyle 1999).
“very similar” nodes, thus representing a continuous trend over time. Because the concept of what is “very similar” can be domain specific, we introduce a user-specified cross-period trend strength parameter $\beta$ 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 both included in the output graph (as determined by the clustering solution and within-period trend strength $\alpha$); and (b) the edge weight is less than or equal to a threshold $\eta$ which depends on cross-period trend strength $\beta$. The edge threshold $\eta$ is calculated by taking the average of weights for all the possible edges among the nodes in two adjacent data partitions (say, partitions $i$ and $i+1$) and adjusting it by the user-specified $\beta$ parameter:

$$\eta_{i,i+1} = \beta \cdot \text{AvgDist}(V_i,V_{i+1}) = \beta \left( \frac{k \sum_{p=1}^{k} \sum_{q=1}^{k} d(v_{i,p},v_{i+1,q})}{k^2} \right).$$

Here $v_{i,p}$ is the $p^{th}$ node in the $i^{th}$ partition and $d(v_{i,p}, v_{i+1,q})$ is the distance between nodes $v_{i,p}$ and $v_{i+1,q}$. The term in parentheses is the average edge weight between partitions $i$ and $i+1$. From the definition of $\beta$ and the relationship between $\beta$ and $\eta$, it is easy to see that lowering $\beta$ will result in more edges being filtered out, leaving only the strongest trends displayed in the graph. We have that $\beta \in [0,1]$ and, similarly to $\alpha$, $\beta$ is a global parameter value for the entire data set. When $\beta = 1$, $\eta_{i,i+1}$ is equal to the average edge weight between partitions $i$ and $i+1$, and therefore only edges with weights below average are included in the graph. This procedure can accommodate a variety of cluster similarity comparison measures, such as average-link and min-link (Duda et al. 2000), as well as different distance metrics $d(x,y)$, including

- Euclidean, i.e., $d(x,y) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2},$
- Manhattan, i.e., $d(x,y) = \sum_{i=1}^{m} |x_i - y_i|,$
- or Chebyshev, i.e., $d(x,y) = \max_{i=1...m} |x_i - y_i|.$

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 Figure 4). Note that if $d(x,y) = 0$ then clusters $x$ and $y$ are identical with respect to the chosen distance metric and cluster similarity measure and, thus, most likely have a very similar makeup.

Figure 5 provides an example temporal cluster graph that was filtered using $\alpha = 0.05$ and $\beta = 0.80$. In the first partition no clusters were removed; however, one cluster was filtered out of each of the second and third partitions. This is apparent because $k_2 = 4$ and $k_3 = 3$ but only three nodes are displayed in the second partition and only two in the third partition. Specifically, a cluster of size three and a cluster of size four were filtered out of the second and third partitions respectively. Similarly, in the fourth data partition two clusters whose sizes $s_1$ and $s_2$ where $s_1+s_2 = 8$ and $s_1, s_2 \leq 6$ were filtered out. The edges displayed represent distance metrics between nodes $x$ and $y$ in adjacent partitions where $d(x,y) \leq \eta$ and $\eta$ was determined using $\beta = 0.80$ and the average of the edges between adjacent partitions. Note that several clear trends are represented by the output graph. For example, the first row of nodes across the graph maintains a very high level of similarity until the cluster in the third partition splits into two clusters in the fourth partition. In later sections of the paper, we provide examples of temporal cluster graphs generated from real-world data.

4 C-TREND IMPLEMENTATION

4.1 C-TREND OVERVIEW

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 main phases: (1) offline preprocessing of the data and (2) online interactive analysis and graph rendering (see Figure 6).

In the preprocessing phase, the data set is partitioned based on time periods, and each partition is clustered using any traditional clustering technique, such as a hierarchical approach (Duda et al. 2000). 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. Based on these data structures, graph entities (nodes and edges) are generated and rendered as a temporal cluster graph in the system output window. In the interactive analysis phase, C-TREND allows the user to modify $k_i$ $(i = 1, ..., t)$, $\alpha$, and $\beta$ on demand in real-time and, as a result, update the view of the temporal cluster graph.

4.2 PREPROCESSING

DATA CLUSTERING AND THE DENDROGRAM DATA STRUCTURE

An important requirement for real-time graph customization in C-TREND is the pre-computation of multiple clustering solutions from the initial data set. Depending on the type of clustering algorithm employed, the cluster solution can be stored in a way that maximizes the effi-
ciency of the output graph customization. Most standard clustering techniques are based on measuring distance between clusters, and there has been extensive research on clustering techniques in the data mining literature (see Duda et al. 2000, Jain et al. 1999, and Kaufman and Rousseeuw 1990 for examples). Since C-TREND is a meta-analysis technique, it can employ different standard clustering algorithms (e.g., agglomerative or divisive hierarchical clustering or partition-based clustering). It can also use different cluster distance metrics, including minimum-link (nearest neighbor), maximum-link (farthest neighbor), average-link, and the distance between cluster centers (as used in our current analysis) as well as different basic distance metrics between individual data points (e.g., Euclidean, Manhattan, Chebyshev). Specifically, for our experiments we utilized agglomerative hierarchical clustering (Jardine and Sibson 1968, Johnson 1967) based on the Euclidean distance between cluster centers, which is performed separately for each partition of the data. Agglomerative hierarchical clustering procedures start with \( n \) singleton clusters (i.e., each cluster is a single data point) and successively merge the two “closest” clusters over \( n-1 \) iterations until one comprehensive cluster is assembled.

C-TREND utilizes optimized dendrogram data structures for storing and extracting cluster solutions generated by hierarchical clustering algorithms. While C-TREND can possibly be extended to support partition-based clustering methods (e.g., \( k \)-means), hierarchical clustering is particularly well-suited for real-time updates because the clustering process has to be performed only once to create a complete set of solutions (which also makes the zoom operation very efficient). Other clustering methods may require the computation of a range of solutions based on different numbers of anticipated clusters.

The clustering solution of each data partition is represented by a dendrogram. Figure 7 depicts an example dendrogram that results from hierarchically clustering a set of ten data points, where at each level the two “nearest” clusters are combined into one. Agglomerative hierarchical clustering on a data partition of size \( n \) would produce a dendrogram with \( 2n-1 \) nodes, \( n \) of which are leaves. The dendrogram therefore contains all \( n \) possible clustering solutions, where the largest clustering solution contains \( n \) singleton clusters and the smallest solution contains a single cluster with all \( n \) data points in it. Additionally, the “height” of each node in the dendrogram represents the order of clustering. C-TREND produces a dendrogram for each data partition and utilizes a global input value \( N \) that represents the maximum-sized cluster solution maintained for each data partition. For all practical purposes, a useful solution will consist of a set of \( N \ll n \) clusters, and, therefore, C-TREND has to store only \( 2N-1 \) nodes per partition. We have found that maintaining a maximum solution size consisting of \( N=50 \) clusters is more than sufficient for many practical applications of data visualization (i.e., visualizing more than 50 clusters at a time in each partition can be overwhelming and counter-productive for the the user). In general, the value of \( N \) can also be set by the user as needed.

To maintain the dendrogram information that results from hierarchical clustering, C-TREND uses an optimized array-based data structure for each data partition. Each array element represents a single node in the partition’s dendrogram. Each node represents a cluster in some clustering solution, and the array includes the corresponding cluster size, the cluster center (a vector of cluster’s average values of data attributes), and the pointers to the left and right children in the dendrogram. For leaf nodes, the left and right children are assigned null values.

This dendrogram data structure allows for quick extraction of any specific clustering solution for each data partition when the user changes partition zoom level \( k \). To obtain a specific clustering solution from the data structure for data partition \( D_i \), C-TREND uses the DENDRO_EXTRACT algorithm (Algorithm 1) which takes the desired number of clusters in the solution, \( k \), as an input and returns the set \( CurrCl \) containing the clusters corresponding to the \( k \)-sized solution. Cluster attributes, such as center and size, are then accessible from the corresponding dendrogram data structure by referencing the clusters in \( CurrCl \).

C-TREND produces a dendrogram for each data partition and utilizes a global input value \( N \) that represents the maximum-sized cluster solution maintained for each data partition. For all practical purposes, a useful solution will consist of a set of \( N \ll n \) clusters, and, therefore, C-TREND has to store only \( 2N-1 \) nodes per partition. We have found that maintaining a maximum solution size consisting of \( N=50 \) clusters is more than sufficient for many practical applications of data visualization (i.e., visualizing more than 50 clusters at a time in each partition can be overwhelming and counter-productive for the user). In general, the value of \( N \) can also be set by the user as needed.

To maintain the dendrogram information that results from hierarchical clustering, C-TREND uses an optimized array-based data structure for each data partition. Each array element represents a single node in the partition’s dendrogram. Each node represents a cluster in some clustering solution, and the array includes the corresponding cluster size, the cluster center (a vector of cluster’s average values of data attributes), and the pointers to the left and right children in the dendrogram. For leaf nodes, the left and right children are assigned null values.

This dendrogram data structure allows for quick extraction of any specific clustering solution for each data partition when the user changes partition zoom level \( k \). To obtain a specific clustering solution from the data structure for data partition \( D_i \), C-TREND uses the DENDRO_EXTRACT algorithm (Algorithm 1) which takes the desired number of clusters in the solution, \( k \), as an input and returns the set \( CurrCl \) containing the clusters corresponding to the \( k \)-sized solution. Cluster attributes, such as center and size, are then accessible from the corresponding dendrogram data structure by referencing the clusters in \( CurrCl \).

C-TREND produces a dendrogram for each data partition and utilizes a global input value \( N \) that represents the maximum-sized cluster solution maintained for each data partition. For all practical purposes, a useful solution will consist of a set of \( N \ll n \) clusters, and, therefore, C-TREND has to store only \( 2N-1 \) nodes per partition. We have found that maintaining a maximum solution size consisting of \( N=50 \) clusters is more than sufficient for many practical applications of data visualization (i.e., visualizing more than 50 clusters at a time in each partition can be overwhelming and counter-productive for the user). In general, the value of \( N \) can also be set by the user as needed.

To maintain the dendrogram information that results from hierarchical clustering, C-TREND uses an optimized array-based data structure for each data partition. Each array element represents a single node in the partition’s dendrogram. Each node represents a cluster in some clustering solution, and the array includes the corresponding cluster size, the cluster center (a vector of cluster’s average values of data attributes), and the pointers to the left and right children in the dendrogram. For leaf nodes, the left and right children are assigned null values.

This dendrogram data structure allows for quick extraction of any specific clustering solution for each data partition when the user changes partition zoom level \( k \). To obtain a specific clustering solution from the data structure for data partition \( D_i \), C-TREND uses the DENDRO_EXTRACT algorithm (Algorithm 1) which takes the desired number of clusters in the solution, \( k \), as an input and returns the set \( CurrCl \) containing the clusters corresponding to the \( k \)-sized solution. Cluster attributes, such as center and size, are then accessible from the corresponding dendrogram data structure by referencing the clusters in \( CurrCl \).

C-TREND produces a dendrogram for each data partition and utilizes a global input value \( N \) that represents the maximum-sized cluster solution maintained for each data partition. For all practical purposes, a useful solution will consist of a set of \( N \ll n \) clusters, and, therefore, C-TREND has to store only \( 2N-1 \) nodes per partition. We have found that maintaining a maximum solution size consisting of \( N=50 \) clusters is more than sufficient for many practical applications of data visualization (i.e., visualizing more than 50 clusters at a time in each partition can be overwhelming and counter-productive for the user). In general, the value of \( N \) can also be set by the user as needed.
Algorithm 1. DENDRO_EXTRACT
INPUT: $k_i$ - desired number of clusters
i - data partition indicator
1 begin
2 if $k_i \leq N$ then
3 CurrCl = {DendrogramRoot}
4 while |CurrCl| < $k_i$
5 $MaxCl = DendrogramRoot, -|CurrCl| + 1$
6 CurrCl = (CurrCl \ $MaxCl$) $\cup$
7 return CurrCl
8 else request new $k_i$
9 end

gram and traverses the dendrogram by splitting the highest-numbered node (where the nodes are numbered according to how close they are to the root, as numbered in Figure 7) in the current set of clusters until $k$ clusters are included in the set. $MaxCl$ represents the highest element in the current cluster set $CurrCl$. It is easy to see that, because of the specific dendrogram structure, it is always the case that $MaxCl = DendrogramRoot, -|CurrCl| + 1$. Furthermore, the dendrogram data array maintains the successive levels of the hierarchical solution in order, therefore replacing $MaxCl$ by its children $MaxCl.Left$ (left child) and $MaxCl.Right$ (right child) is sufficient for identifying the next solution level in the dendrogram. DENDRO_EXTRACT is linear in time complexity, $O(k)$, which provides for the real-time extraction of cluster solutions.

Example: Consider C-TREND data partition $D_i$, whose clustering solution is represented by the dendrogram in Figure 7. Let the desired clustering solution for this partition consist of four clusters (i.e., $k_i = 4$). To display this solution, the DENDRO_EXTRACT algorithm will start with the root node, cluster 19, and set $CurrCl = \{19\}$ (line 3). Since $k_i = 4$ and $CurrCl$ only contains one cluster (line 4), DENDRO_EXTRACT will continue through the while loop (lines 5-6). Initially, $MaxCl = 19$ because it is the highest-numbered (and the only) cluster in the $CurrCl$ set (line 5), therefore $CurrCl$ is updated by replacing cluster 19 by its children nodes, clusters 16 and 18 (line 6). In the second iteration, $|CurrCl| = 2$ and $MaxCl = 18$, and cluster 18 is replaced by its children, clusters 6 and 17. In the next iteration, $|CurrCl| = 3$ and $MaxCl = 17$, which is replaced by its children, clusters 11 and 12. At this point, $|CurrCl| = k_i = 4$ and the algorithm outputs the solution for $k = 4$, which is $CurrCl = \{6, 11, 12, 16\}$.

**OPTIMAL NUMBER OF CLUSTERS**

To set an initial value $k_i$ for each data partition $D_i$, C-TREND uses the “optimal” number of clusters based on the so-called “elbow” (or “gap”) criterion for comparing the fitness of different cluster solutions. We use mean squared error (MSE) as our fitness metric (Duda et al. 2000). In agglomerative hierarchical clustering, $n$ solutions are created containing $1, \ldots, n$ clusters respectively. C-TREND determines the largest “jump” in MSE across these $n$ solutions, which points to the optimal number of clusters. In a plot of the MSE, this jump would be represented by a sharp elbow (i.e., a significant “flattening” in the solution fitness increase).

The “elbow” criterion is discussed in further detail in (Aldenderfer and Blashfield 1984, Sugar 1999, Sugar and James 2003), and the related “gap” statistic is discussed in (Tibshirani et al. 2001). For example, the basic method for determining the optimal number of clusters used by Sugar and James (2003) is to: (a) calculate a distortion metric for several $k$-sized clustering solutions (i.e., for different $k$ values); (b) transform the distortion metric based on the data dimensionality; and (c) determine the largest jump in the transformed distortion metric, which indicates the optimal number of clusters. In our experiments, we used MSE as the transformed distortion metric, but this is just one possible solution. Another common method for determining the optimal number of clusters is to use the amalgamation coefficient (Aldenderfer and Blashfield 1984). Also, Milligam and Cooper (1985) provide a review of several other metrics used to determine the optimal number of clusters for a clustering solution, and C-TREND can be extended to support any of these alternative approaches.

**NODE LISTS AND EDGE LISTS**

The last step in preprocessing is the generation of the node list, which contains all possible nodes and their sizes, and the edge list, which contains all possible edges and their weights, for the entire data set. Creating these lists in the preprocessing phase allows for more effective (real-time) visualization updates of the C-TREND graphs.

As mentioned above, each data partition possesses an array-based dendrogram data structure containing all its possible clustering solutions. The node list is simply an aggregate list of all dendrogram data structures indexed for optimal look-up of nodes.

The edge list is generated based on the node list, since an edge is possible between any two nodes in adjacent data partitions. Therefore, the edge list is essentially a list of ordered pairs, where each pair represents adjacent nodes that define an edge. Since each dendrogram contains $2N-1$ nodes (i.e., $N$ leaves and $N-1$ internal nodes), there are $(2N-1)^2$ possible edges between adjacent data partitions. If the data set possesses $t$ data partitions, then there are $(t-1)^2(2N-1)^2$ possible edges for the entire data set. Note that we use $t-1$ because the first partition only possesses outgoing edges and the last partition only possesses incoming edges. Therefore, the time complexity of the edge list generation would have an asymptotic upper bound of $O(N^2tm)$, where $m$ is the number of attributes in the data. By 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.

Table 1 shows the preprocessing time required to generate edge lists of varying size. It is easy to see that, even in some extreme cases where $N=100$ or 500, edge list preprocessing takes less than a minute. These experiments were performed by implementing the edge list generation procedure in C programming language and testing it on a fairly typical desktop PC with a Pentium 4 3.4 GHz proc-
It should be noted that the results reported in Table 1 were calculated holding the number of attributes in the data constant at 10. Since this process requires the calculation of a distance metric for each edge, the time it takes to generate the edge list should increase linearly with the number of attributes in the data. To demonstrate that this is indeed the case, Figure 8 contains a plot of the increase in edge list generation time as the number of attributes is being increased from 10 to 100, holding N and t constant.

4.3 INTERACTIVE DATA VISUALIZATION

C-TREND utilizes a series of validation flags to maintain and update the displayed state of the output trend graph. Combinations of the validation flags are used to determine whether or not each possible edge and node should be displayed in the graph, and, as these flags change, the displayed components of the graph also change.

Each cluster in the node list (dendrogram data structures) possesses two flags: $k$-pass and $\alpha$-pass. These flags are used to indicate whether the cluster should be included in the output graph based on the $k$-value and the $\alpha$-value, respectively. Specifically, when $k_i$ is changed, the dendrogram data structure is updated so that only the clusters that should be extracted for the clustering solution of size $k_i$ have a valid $k$-pass flag. Similarly, when $\alpha$ is changed, the dendrogram data structure is updated so that only the clusters that are large enough to pass the node filter based on $\alpha$ are assigned a valid $\alpha$-pass flag. The nodes that have both valid $k$-pass and $\alpha$-pass flags make up the set of nodes that are both large enough and in the desired clustering solution and, therefore, are included in the output graph.

In our implementation, a list of all possible edges and their weights is generated on the initial load of the data. To demonstrate that this process depends on three basic operations: changing any one parameter requires only one pass

Each of these operations requires a set of calculations to be performed on the node and edge lists, and we will show that, for most practical purposes, these operations are performed efficiently enough to provide “real-time” graph generation and modification. By implementing the $k$-pass, $\alpha$-pass, and $\beta$-pass validation flags we create some independence in the parameters. The parameters can be adjusted independently, and graph elements are rendered only if they are valid based on the status of the flags. As flags are changed and elements become valid, they are rendered in real time.

Computational complexity of the parameter-changing operations can be easily calculated based on the maximal number of clusters $N$ and the number of data partitions $t$. Changing $\alpha$ requires C-TREND to scan through all possible nodes to determine if each node should have a valid or invalid $\alpha$-pass flag. If each partition contains $2N-1$ possible nodes and there are $t$ partitions, changing $\alpha$ has an asymptotic upper bound of $O(Nt)$. Similarly, $\beta$ is also a graph-wide parameter, and changing $\beta$ requires C-TREND to scan through all possible edges to determine if each edge should possess a valid or invalid $\beta$-pass flag. Since there are $(t-1)(2N-1)^2$ possible edges, changing $\beta$ is $O(N^2t)$.

Changing $k_i$ is extremely efficient, since each data partition $D_i$ has its own corresponding $k_i$ value, and the operation simply updates the $k$-pass flags for clusters in the corresponding data partition. This operation uses the DENDRO_EXTRACT procedure (Algorithm 1) described in Section 3. Recall that DENDRO_EXTRACT iterates $k_i$ times until it has found the $k_i$ clusters that make up the $k_i$-sized clustering solution, where each iteration needs only constant-time operations – thus, DENDRO_EXTRACT takes $O(k_i)$ time. Changing $k_i$ simply uses DENDRO_EXTRACT to update the $k$-pass flags of the data structure and, because each partition has at most $2N-1$ clusters, changing $k_i$ would require at most $2N-1$ operations. Therefore, in the worst case, changing $k_i$ is linear in the maximum sized clustering solution, $O(N)$.
To optimize changing $k_i$ even further, C-TREND uses simple algorithms to increment and decrement the $k_i$ value by one. Recalling the notation from Section 3, to increase the number of clusters included in a partition by one (i.e., $k_i := k_i + 1$), the MaxCl in the set of current clusters, CurrCl, for the $k_i$-sized solution is replaced by its children MaxCl.Left and MaxCl.Right: $\text{CurrCl} = (\text{CurrCl} \setminus \text{MaxCl}) \cup (\text{MaxCl}.\text{Left} \cup \text{MaxCl}.\text{Right})$. A very similar approach can be applied to decrement the number of clusters included in a partition: $\text{CurrCl} = (\text{CurrCl} \cup \text{NextCl}) \setminus (\text{NextCl}.\text{Left} \cup \text{NextCl}.\text{Right})$, where NextCl is the cluster that is the next highest in the dendrogram hierarchy after MaxCl. These simple optimizations that exploit the dendrogram data structure reduce the complexity of changing $k_i$ operation to $O(1)$ for increments and decrements of $k_i$.

To validate theoretical efficiencies discussed above, we implemented C-TREND in C programming language and measured the time required to perform several operations on a fairly typical desktop PC with a Pentium 4, 3.4 GHz processor with 1 GB of RAM. Table 2 contains the results of this analysis. We performed the analysis on six data sets ranging in the number of data partitions ($t$) and the maximum-sized cluster solution ($N$) for each partition. For each experiment, we ran the operation for five different randomly generated data sets, and Table 2 displays the average time to perform each operation. Notice that the changing $k_i$ and incrementing/decrementing $k_i$ operations took less than 1 $\mu$s regardless of the size of the data set used. Additionally, changing $\alpha$ also took less than 1 $\mu$s for each dataset analyzed. Only the changing $\beta$ operation required processing times greater than 1 $\mu$s. For the data set with a maximum sized clustering solution of 500, changing $\beta$ took 0.192s. Recall that the changing $\beta$ operation has the computational complexity of $O(N^2)$, and that for most practical purposes it is sufficient to have $N \leq 50$. This suggests that changing $\beta$ will definitely take less than 0.084s for typical uses of C-TREND (the last row in Table 2 shows results for an even more complex situation where $N=100$ and $t=100$). Additionally, the increase in processing time for these operations from 0.007s to 0.192s corresponding to the increase of $N = 100$ to $N = 500$ agrees with our theoretical calculations of $O(N^2)$. The experimental results in Table 2 support our claims that C-TREND output graph generation and modification can be done in real time and can provide instant visual feedback to the user when parameters are changed. We should also note that, unlike the edge list generation procedure in preprocessing, none of the parameter changing operations will be dependent on the number of data attributes. This is because changing parameters requires only simple lookups and comparisons, and no distance metric calculations are required.

To further demonstrate the scalability of the C-TREND technique, we report a second set of experimental results in Figures 9 and 10. We demonstrate the efficiency of C-TREND in its most computationally costly operation, updating $\beta$, for various large data sets. In Figure 9, we show that, while increasing the maximum sized clustering solution $N$, C-TREND can still update $\beta$ relatively quickly. In fact, for a data set with 150 data partitions and $N = 200$, $\beta$ is updated in less than 0.6 seconds. Figure 10 demonstrates that a dramatic increase in the number of data partitions can also be handled by C-TREND in a real-time fashion. In fact, for a very large data set containing 10,000 partitions with $N=50$, $\beta$ is updated in less than 2 seconds.

### 5 Evaluation: Case Study on Wireless Networking Certifications

C-TREND and temporal cluster graphs provide a versatile technique for identifying and representing trends in temporal multi-attribute data. As mentioned earlier, two important parameters are used to filter out spurious graph entities: within-period trend strength $\alpha$ and cross-period trend strength $\beta$. Temporal cluster graphs can be based on multiple clustering methods, distance metrics, and cluster similarity measures. To demonstrate the use of temporal cluster graphs for identifying trends in real-
world transactional attribute-value data, we present the analysis of over 2,400 certifications for new wireless networking technologies based on the IEEE 802.11 standard, that are 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. Products can be certified based on a number of standards, 15 in all, including IEEE protocol (802.11a, b, g, d, and h), security protocol (e.g., WPA-personal, WPA-enterprise, and WPA2), authentication protocol (e.g., EAP, EAPTLS, and PEAP), and quality of service (e.g., WMM and WMM Power Save). Each product certification consists of a date of certification and a set of binary attributes indicating the presence or absence of the standards listed above.

The Wi-Fi certifications data set is a good example for a proof of concept of the temporal cluster graph. The data is multi-attribute and temporal, and analysts interested in the evolution of wireless networking technologies can use temporal cluster graphs to identify trends in product types over time. For our analysis, the certification data included all standards-related attributes as well as the product type (e.g., PDA, internal card) and product category (i.e., whether it is a component, device, or infrastructure technology) attributes. Certifications were coded into product categories based on the similarity of their product type and functionality; for example, compact flash adapters, internal cards, external cards, and USB clients were grouped into the component category, because they all act as components that provide Wi-Fi functionality to other products, such as PCs or laptops.

Figures 11 and 12 present trend graphs for the Wi-Fi data partitioned into one-year time periods. For each time period, a set of clusters has been identified as nodes. Each node is labeled with the size of the cluster and can be intuitively described by its center. For example, in Figure 11 the cluster labeled 82 in 2001 contains 82 data points and has a center vector (1, 0, 0, 0, 0, 0.01, 0, 0, 0.46, 0.38, 0.15, 0, 0, 0, 0, 0.04, 0.04, 0, 0, 0.04, 0, 0, 0, 0, 0, 0). It should be noted that all attributes for this data set were binary (1 if the product possessed the attribute and 0 otherwise), and, therefore, the center values indicate that this cluster is made up of 100% components with 1% compact flash cards, 46% internal cards, 38% external cards, and 15% USB client devices. Of these components, 100% are 802.11b-certified and 4% have WPA-personal, WPA-enterprise, and EAPTLS certifications. Based on this information, it is clear that all of the Wi-Fi components (which are mostly cards) are clustered together at this point in the timeline.

Edges were rendered between nodes in adjacent time periods to represent similarities between clusters over time. For example, the edge labeled 0.08 in Figure 11 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, the next edge has a weight of 0.04. Therefore, by looking at a temporal cluster graph the user can see 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, however, we see a larger deviation with an edge weight of 0.26 which indicates a reduction in similarity. The trend continues to 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 as indicated by the similarity measure of 1.6. However, this distance is still

2 From Wi-Fi.org: “The Wi-Fi Alliance is a global, non-profit industry trade association with more than 200 member companies devoted to promoting the growth of wireless Local Area Networks (WLAN). Our certification programs ensure the interoperability of WLAN products from different manufacturers, with the objective of enhancing the wireless user experience.”

3 In this paper, we do not provide such detailed information within the graph figures themselves because of the space limitations; however, this information is readily accessible to C-TREND users (e.g., by clicking on any node in the graph).
less than the threshold for edge weights, which is the average of all edge weights between the two partitions multiplied by $\beta = 0.75$. Specifically, the C-TREND tool provides an intuition that the introduction of 802.11g and security technologies led to a significantly different class of wireless access points in 2004.

Some other clear trends in wireless networking technologies are also visible in Figure 11. A trend of wireless network cards converges in 2001 and continues into 2003 when 802.11g is introduced. 802.11b embedded laptop clients first appear in 2003, and in 2004 internal cards enabled with all possible technical attributes appear (802.11b/g/a/h/d and all security/QoS specifications).

By comparing Figures 11 and 12 we can see the effect of modifying the zoom-level and the $\alpha$ and $\beta$ parameters. Figure 11 presents a trend graph using $\alpha = 0.02$ and $\beta = 0.75$, which excludes clusters smaller than 2% of the total number of data points in a data partition and edges more than 75% of the average edge weight. Figure 12 has more relaxed parameters with $\alpha = 0.0$ and $\beta = 1.0$ and, therefore, includes many additional nodes and edges. The advantage of adjusting $\alpha$ and $\beta$ is that it provides the C-TREND user the ability to show or hide possible trends according to their strength. For example, in Figure 12 the more relaxed parameters allow the C-TREND users to uncover a new cluster of size 6 in 2002 and identify trends that were not apparent in Figure 11. On the other hand, Figure 12 includes many clusters of size 1 with either no adjoining edges or edges with high weights. 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, as in Figure 11. The zoom-level of the 2001 partition was set to seven clusters in Figure 12 and five clusters in Figure 11. Additionally, the the zoom-level of the 2003 partition was set to five clusters in Figure 12 and seven clusters in Figure 11. Adjusting the zoom-level allows the user to apply her domain expertise to select the optimal granularity for displaying the most relevant clusters.

The Wi-Fi technology analysis demonstrates the ability of temporal cluster graphs to identify and represent trends in multi-attribute data. One would expect the Wi-Fi data to contain clear trends based on the versions of the 802.11 standard and the technical features. C-TREND produced temporal cluster graphs that correctly identified these changes in the data, presented them in an intuitive and useful manner, and also provided additional insights. One could use C-TREND to further explore the Wi-Fi trends using different time windows and attribute sets (not provided here because of the space limitations).

6 DISCUSSION

6.1 TREND METRICS

As demonstrated above, temporal cluster graphs provide a novel approach for identifying and visualizing trends in multi-attribute transactional data. These graphs help the user visualize relationships between dominant transaction types in a data set over time. To provide additional analytical power to the users, we next discuss a set of trend metrics that allow the users to analyze the trend directionality as well as other trend characteristics and, as a result, better understand the significance and scale of the patterns identified.

Before we define the trend metrics, we first present the some notation. As before, let $G = (V, E)$ be a temporal cluster graph where $V$ is the set of all nodes and $E$ is the set of all edges in the graph. Trend $P$ of length $n$ is defined as a path of $n$ edges, containing $n+1$ nodes in graph $G$. In other words, $P = (v_0, ..., v_n)$, where $e(v_{i+1}, v_i) \in E$ for $i = 1, ..., n$. The distance between nodes $x$ and $y$ is defined as $d(x, y)$ and, as previously discussed, can be measured using Euclidean distance between $x$ and $y$ (or any other distance metric), where each node $x$ is represented by a corresponding cluster center, $x = (x_1, ..., x_m)$; here $m$ denotes the number of attributes (dimensions) in the data.

Therefore, the distances between nodes represented by edge weights in the output graph are the distances between the actual clusters in multidimensional space. Below we present several trend metrics that are general and can be used with a variety of cluster distance metrics and cluster similarity measures.

Path distance, $d_P$, measures the total distance traveled along a trend (path) and is defined as

$$d_P(v_s, v_e) = \sum_{i=1}^{n} d(v_{i-1}, v_i).$$

In other words, path distance is the sum of the weights of all edges within a path. Direct distance, $d_D$, measures the absolute distance between the head (first) and tail (last) nodes in a path and is defined as

$$d_D(v_s, v_e) = d(v_s, v_e).$$

Transitive distance, $d_T$, measures the maximum direct distance between any two nodes in the path and is defined as

$$d_T(v_s, v_e) = \max_{v_{i,j}} d(v_s, v_j).$$

Note that the transitive distance does not necessarily have to correspond to an edge that exists in the output graph.

It is easy to see from their definitions, that the three distances introduced above have the following property: $0 \leq d_D(v_s, v_e) \leq d_P(v_s, v_e) \leq d_T(v_s, v_e)$. These distances provide useful information about paths in a graph. The path distance indicates the cumulative amount of change in a multi-attribute space along a trend; the direct distance provides a comparison between the starting and ending points of the trend; and the transitive distance indicates the maximum difference in clusters within a trend.

The information provided by these distances should prove useful to the analyst, and taking various ratios of these distances provides even more detail about the nature of a trend. For instance, the trend directionality ratio, $dir(v_s, v_e)$, is defined as the ratio of the direct distance and the path distance:

$$dir(v_s, v_e) = \frac{d_D(v_s, v_e)}{d_P(v_s, v_e)}.$$

Because $0 \leq d_D(v_s, v_e) \leq d_P(v_s, v_e)$, we have that $dir(v_s, v_e) \in [0, 1]$. Specifically, the directionality ratio measures the
continuity of consistent directional change in a trend. A value of $\text{dir}(v_0, v_n) = 1$ means that $d_d(v_0, v_n) = d_s(v_0, v_n)$ and, therefore, changes in the cluster centers were consistently in the same direction throughout the trend for every attribute. Alternatively, $\text{dir}(v_0, v_n)$ that is close to zero indicates that $d_d \ll d_s$. In other words, the trend is not moving consistently in the same direction through multidimensional space. Figure 13 depicts two scenarios for illustrating different directionality ratios for path $P = (c_1, c_2, c_3, c_4)$, where clusters $c_1, c_2, c_3,$ and $c_4$ would appear as nodes in partitions $t_1, t_2, t_3,$ and $t_4$ of a graph, respectively. From the figure it is clear that $\text{dir}(c_1, c_4)$ 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.

A second ratio of interest is the trend expansion ratio, $\text{expn}(v_0, v_n)$, which is defined as the ratio of the transitive distance and the path distance:

$$\text{expn}(v_0, v_n) = \frac{d^T(v_0, v_n)}{d^P(v_0, v_n)}.$$

Similar to the directionality ratio, $\text{expn}(v_0, v_n) \in [0, 1]$. The expansion ratio is useful for determining the magnitude of directional change in a trend and, therefore, it complements the directionality ratio. Since the transitive distance measures the maximum absolute distance between any two clusters on a path, a larger transitive distance indicates a longer consistent movement in the same direction within a path. The expansion ratio measures this consistency with respect to the total path distance. Paths with longer periods of directional change will have higher expansion ratio values while paths that “wind” back and forth will have lower expansion ratios. Figure 14 provides an example of two scenarios for illustrating different expansion ratios for path $P = (c_1, c_2, c_3, c_4)$. Note that the directionality ratios for scenario 1 and scenario 2 in Figure 14 would be fairly similar, but comparing expansion ratios captures the difference in path shape. In scenario 1, the transitive distance is about half of the path distance, indicating a longer period for directional change and, therefore, a higher expansion ratio. In scenario 2, the transitive distance is much smaller than in scenario 1, indicating a shorter period for directional change (a winding path) and therefore a lower expansion ratio.

The C-TREND implementation provides path ($d_P$), direct ($d_d$), and transitive ($d^T$) distance metrics as well as trend directionality and expansion ratios for any path selected by the user through a graphical user interface.

### 6.2 Applications

In the previous section we provided a detailed case study demonstrating the use of the temporal clustering technique and the C-TREND implementation to analyze the evolution of wireless networking technologies. While the proposed technique was well suited for the Wi-Fi Alliance certification data, it is designed to be used with any set of multi-attribute data that can be partitioned and clustered. To demonstrate this, we next present a brief analysis of Major League Baseball (MLB) batting statistics using temporal cluster graphs. The data used for this analysis was obtained from Sean Lahman’s Baseball Archive, available for download at http://baseball1.com. Yearly batting statistics were collected for every MLB player that played in the years 1967-2006. For the analysis, four attributes were used: hits, home runs, strike outs, and walks. Each attribute was first normalized by the number of at bats (the number of batting appearances) and then normalized by the range of each variable. Batting records with less than 100 at bats and zero home runs were removed to reduce the skewness of attribute distributions. The data was partitioned into eight five-year subsets (see Figure 15).

The analysis of the partitioned data reveals some interesting trends in the baseball batting data. First, there is a strong trend over the years for average hitters. The performance of average hitters has not changed much over the past 40 years and this is indicated by the very small distances between clusters in adjacent time partitions.

---

4 Note that Figures 13 and 14 present distances between clusters in multidimensional attribute space (in this case two-dimensional, for more intuitive visualization) and are not temporal cluster graphs.
Another interesting trend is periodic appearance of clusters of power hitters (i.e., hitters with significantly more home runs). Additionally, sub-par hitters are apparent in the early years of the data set, but are either absorbed by other clusters or are not as prevalent in later years.

Figure 15 also displays trend metrics calculated for all paths between the node labeled “1338” in the first partition and the node labeled “1668” in the fourth partition. Three possible paths exist between these nodes; P1 is 1338→1379→1560→1668, P2 is 1338→48→1560→1668, and P3 is 1338→42→1560→1668. The direct distance is obviously the same for all paths ($d_0 = 0.03$), however the path distance for path 1 is $d_1 = 0.09$ which is significantly smaller than the other two paths. This indicates that the total distance traveled along path 1 is shorter than the other paths. Additionally, the directionality ratio for path 1 is $dir(P_1) = 0.33$ as compared to $dir(P_2) = 0.05$ and $dir(P_3) = 0.04$. This suggests that the changes in cluster center location along P1 are more consistently in the same direction as compared to P2 and P3. The directionality ratio for P1 suggests that there has been little change in the average batter’s statistics over the past 45 years. Further, the expansion ratios for each path are $expan(P1) = 0.55$, $expan(P2) = 0.48$, and $expan(P3) = 0.51$. This suggests that all three paths contain about the same amount of directional change.

Finally, all of the temporal cluster graphs shown thus far have used data partitioned along the temporal dimension. It should also be noted that it is possible to use the temporal cluster graphs and the C-TREND application to analyze data partitioned along other dimensions. Any continuous or ordinal categorical attribute can be used as the reference for data partitioning. As a proof of concept, we partitioned the same baseball batting data based on the weight of the player. Eight groups of players were defined based on equally-sized partitions, four on each side of the mean player weight. Nodes, edges, and trend metrics were derived in the same manner as was shown for the temporal partitioning. The resulting graph allows the user to identify and analyze additional trends in the batting statistics based on players’ weight as shown in Figure 16. In other words, we can see trends in batting statistics as we move from lightweight players to the heavyweight ones. For example, one can notice that there are much fewer above-average and below-average lightweight hitters in terms of their batting performance.

6.3 MANAGERIAL SIGNIFICANCE, LIMITATIONS, AND FUTURE RESEARCH DIRECTIONS

In this section, we have demonstrated some of the possible applications of temporal cluster graphs using two very different data sets, the Wi-Fi data and baseball data. However, this technique can be applied in a wide variety of data analysis settings. For example, in business applications, the C-TREND system can also 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 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.

This work provides many directions for future research. Temporal cluster graphs provide a general framework for developing new trend analysis techniques. We plan to develop additional functionality in the C-TREND system by extending the set of metrics for measuring trend and graph characteristics. We will focus on such issues as measuring trend strength, comparing trends, and interpreting graph structure. Additionally, we plan to implement an automated mechanism for partitioning a data set. At present, data partitions are defined exogenously by the user; however, it may be advantageous to use data mining techniques to identify optimal data partitions. Furthermore, temporal cluster graphs provide the initial structure for developing predictive models and hypotheses for the existence, birth, death, and continuation of trends in data.

7 CONCLUSION

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

The authors would like to thank the Digital Technology Center and the Carlson School of Management, University of Minnesota, for providing joint financial support of this research. The authors would also like to thank Prasad Sriram for assistance with the development of the GUI.

REFERENCES


