

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

Gediminas Adomavicius, Jesse Bockstedt

Department of Information and Decision Sciences
Carlson School of Management
University of Minnesota
{ gedas@umn.edu, jbockstedt@csom.umn.edu }

Last Modified: February 1, 2007

Abstract. We introduce C-TREND, a new temporal data analysis and visualization technique for representing trends in multi-attribute transactional data. C-TREND performs meta-analysis on standard agglomerative hierarchical clustering solutions for multiple time period data partitions to generate trend graphs. C-TREND uses nodes to represent commonly recurring transaction types in each period and edges to represent cross-period relationships between the common types. In this paper, we provide an overview of the C-TREND technique, discuss the procedures for node and edge discovery from temporal data, discuss the implementation and performance of C-TREND, and demonstrate an application of the technique by analyzing the change in technical characteristics of wireless networking technologies over five years.

1. Introduction

Business intelligence applications represent an important opportunity for data mining techniques to help firms gather and analyze information about their performance, 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 introduce 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 user leverage their domain expertise to generate customizable knowledge visualization diagrams by adjusting several aggregation and filtering parameters used in analysis.

The research field of data mining has developed many sophisticated methods for

identifying patterns in data in order to provide insights and decision support to users. Identifying *temporal* relationships (e.g., trends) in data constitutes an important problem that is relevant in many business, scientific, and academic settings, and the data mining literature has provided analytical techniques for some specialized types of temporal data analysis, such as, time series analysis (Brockwell and Davis 2001, Keogh and Kasetty 2003, Roddick and Spiliopoulou 2002) and sequence analysis (Pei et al. 2004, Zaki 2001) techniques. However, temporal data can take many forms, most commonly being general event data (i.e., *multi-attribute*) or transactional data with some form of timestamp, for which time series or sequence analysis methods are not particularly well suited. The ability to identify trends in such general temporal data can provide significant benefits, such as competitive advantage to a firm performing forecasts or making decisions on future investments and strategies.

The main contribution of this paper is a novel graph-based data representation and visualization technique for multi-attribute temporal data and its trends over time. This technique uses several intuitive parameters (such as time period size, within-period trend strength, cross-period trend strength, and zoom level) that provide comprehensive control over data visualization to the users through flexible aggregation and filtering operations.

The remainder of this paper is organized as follows. We provide an overview of the basic C-TREND technique in Section 2. We discuss the data structures and preprocessing used by C-TREND in Section 3. In Section 4 we discuss how C-TREND produces trend graphs and provides interactive data visualization and trend representation. We then discuss the performance, optimization, and scalability of C-TREND in Section 5. We illustrate the capabilities of C-TREND by identifying temporal patterns and trends in Wi-Fi technologies using a dataset of approximately 2400 Wi-Fi Alliance (wi-fi.org) product certifications in

Section 6. Finally, we provide conclusions, possible extensions, and discussion of future work in Section 7.

2. Overview of C-TREND

C-TREND is a meta-analysis technique for use with existing data-mining techniques. The C-TREND technique consists of two main processes, offline preprocessing of the data and online interactive analysis and visualization of the trends, as shown in Figure 1.

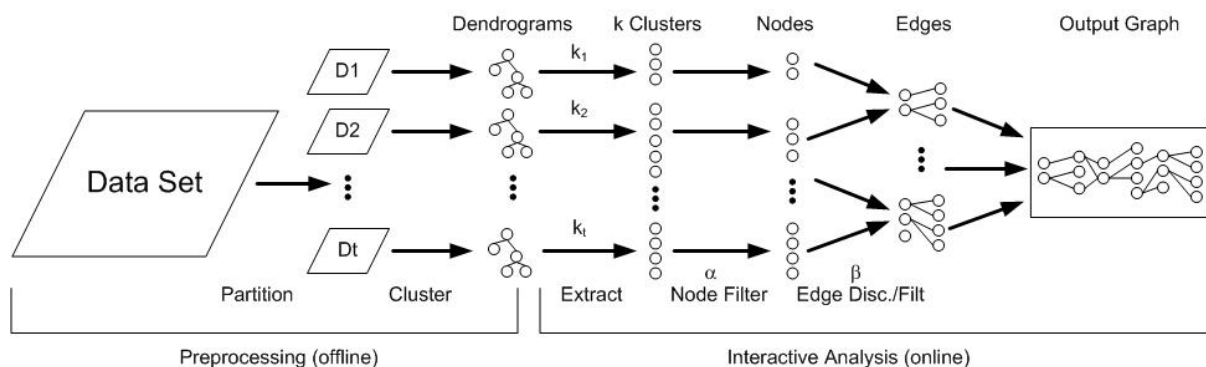


Fig. 1. Overview of C-TREND technique.

Preprocessing consists of dividing and clustering the data. First, a temporal data set is separated into data partitions corresponding to different time periods. Data partitions can be of any size; for example, they can be based on equal time length (e.g., six months), or equal number of data points per partition, or varied. For our case study experiments (presented in Section 6), and without loss of generality, we chose to use equal length time partitions. Second, each data partition is clustered to produce a set of possible clustering solutions of different sizes (i.e., data clustering solutions with 1 cluster, 2 clusters, 3 clusters, and so on). Data mining research has provided many options for clustering data (Duda et al. 2000); specifically, we employed agglomerative hierarchical clustering in our implementation of C-TREND. Agglomerative hierarchical clustering allows using a dendrogram data structure (Duda et al. 2000) which

provides efficiencies in implementation, as will be discussed in Sections 3-5. However, it should be noted that other clustering methods, such as k-means, could also be used (Duda et al. 2000). For each data partition D_i ($i = 1, \dots, t$), the optimal number of clusters is calculated which specifies the size (denoted as k_i) for the initial clustering solution. In the interactive analysis phase, users can then adjust the size of the clustering solution k_i (i.e., number of clusters) for any partition D_i as they see fit. We call k_i the “zoom level” of partition D_i .

C-TREND produces output graphs by arranging the partition clustering solutions chronologically and then identifying similarities between adjacent partitions. In the output graph, each cluster is represented by a node that is described by its size and center. The size of a node (cluster) is the number of data points that make up the cluster, and the center is a vector of the average values of the attributes for each data point in the cluster (i.e., the center represents the “average” data point in the cluster). C-TREND compares every cluster in a data partition to every cluster in an adjacent data partition and calculates a cluster similarity metric. In our experiments, Euclidean distance between cluster centers was used as the cluster similarity metric (i.e., a larger Euclidean distance equates to more dissimilarity between clusters); however, other metrics could also be used. This similarity metric then represents a weight label for edges drawn between nodes in adjacent time periods. By connecting similar nodes with edges across all adjacent data partitions, C-TREND produces an output trend graph as depicted in Figure 2. Figure 2 represents the evolution of Wi-Fi technologies based on product attribute changes over a six year period. The node labels represent the size of the node and the edge labels represent the distance between connected nodes. We provide a detailed discussion of C-TREND output graphs in a case study in Section 6.

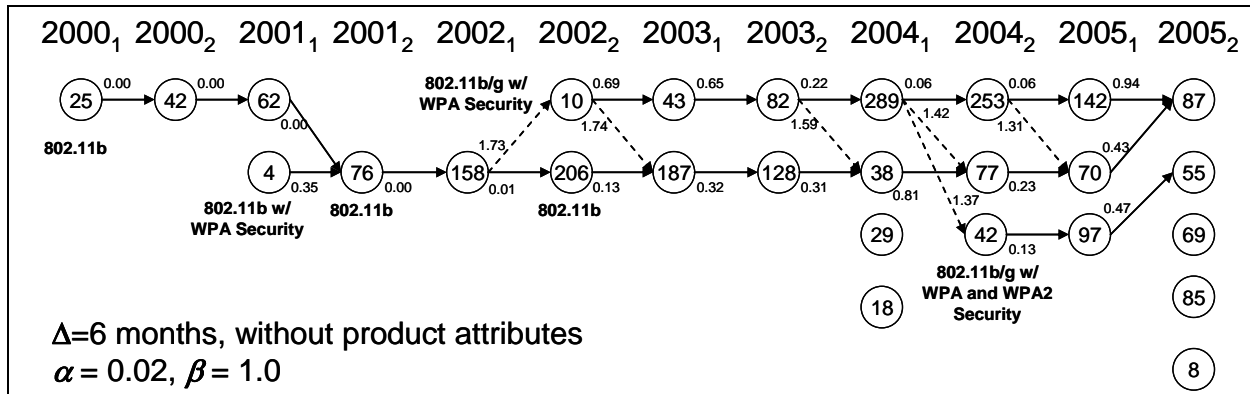


Fig. 2. Wi-Fi technology trend graph

Using interactive analysis and trend visualization of C-TREND, users can adjust the zoom level of any partition (k_i) as well as two graph-level parameters: the within-period trend strength (α), and the cross-period trend strength (β). These parameters serve as thresholds for determining which nodes and edges should be included in the graph. Specifically, within-period trend strength α denotes a minimum size for nodes that should be included, and cross-partition trend strength β denotes the maximum weight for cross-partition edges that should be included. In other words, nodes (clusters) that are too small would be considered spurious, and edges that represent significant dissimilarities between nodes would not signify a temporal trend. C-TREND redraws the graphs in real-time, whenever any of these parameters are interactively adjusted by the user. The power of this approach comes from its ability to represent and interactively explore trends in multi-attribute data using a combination of data mining, graph-theoretic, aggregation, and filtering techniques to aid the data visualization. C-TREND provides a significant amount of control over the analysis and visualization to the user, which is important in many business intelligence applications where domain knowledge is crucial and, therefore, the interactive participation of the domain expert is desired. Section 4 provides more details on the user-adjustable parameters of C-TREND.

As mentioned earlier, C-TREND is designed to work with *transactional attribute-value data*. Specifically, transactional attribute-value data is a general form of temporal data that consists of a collection of records (e.g., stored in a relational table) each with a time (and/or date) stamp and described by a set of attributes. Examples of this type of data would be shopping cart data with a sale timestamp and numerical values representing the number of products purchased in certain categories (e.g., dairy, meat, produce), or product description data that includes a release date and a set of indicators representing the presence and/or quantity of specific product features (e.g., for a laptop computer: USB port, SD card slot, monitor output).

The next two sections of the paper will provide a more comprehensive overview of the two main stages of the C-TREND technique: preprocessing (Section 3) and interactive data visualization (Section 4).

3. Preprocessing and Data Structures

3.1 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 efficiency 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 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, Minkowski). Specifically, for our experiments we utilized agglomerative hierarchical clustering (Jardine and Sibson 1968, Johnson 1967) based on the Euclidean distance between cluster means, 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 an optimized dendrogram data structure for storing and extracting cluster solutions generated from hierarchical clustering algorithms. While C-TREND can easily be extended to support partition-based clustering methods, such as k -means, hierarchical clustering is particularly well suited for real-time updates because the clustering process needs only be performed once to create a complete set of solutions. Other clustering methods, such as k -means, may require the computation of a range of solutions based on an initial number of anticipated clusters.

The clustering solution of each data partition is represented by a dendrogram. Figure 3 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 set 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 solutions contains a single cluster with all n data points in it. 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 $N \ll n$ clusters. 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. In general, the value of N can also be set by the user as needed.

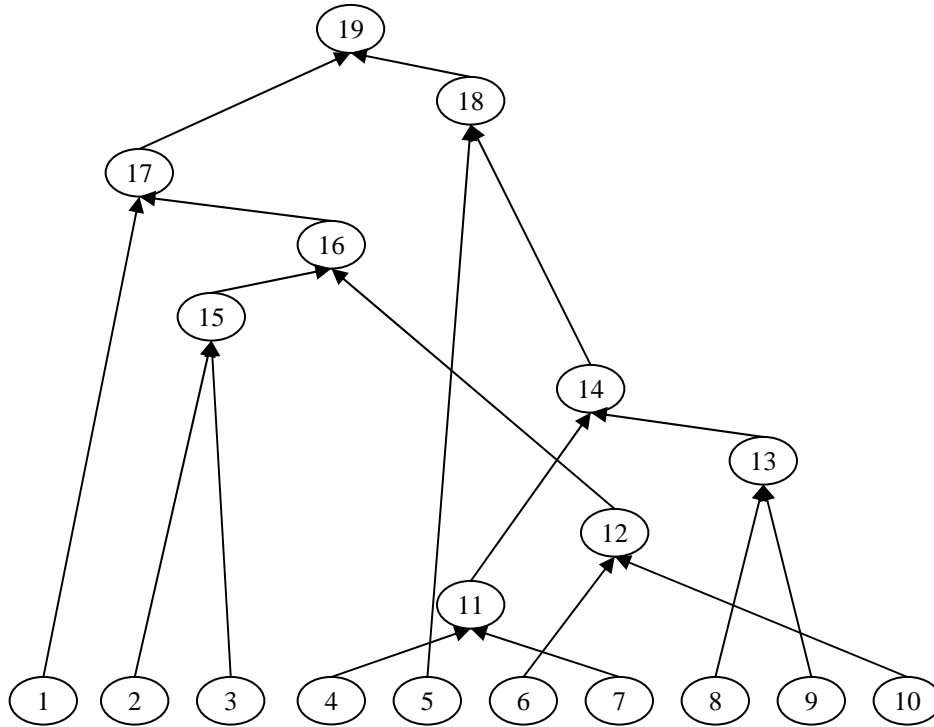


Fig. 3. Example dendrogram with $N = 10$.

To maintain the dendrogram information that results from hierarchical clustering, C-TREND uses an optimized data structure which consists of a list of arrays 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 the data attributes), the radius measure of the cluster (average distance of each point in the cluster to the center), 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. 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_i , as an input and returns the set $CurrCl_i$ containing the clusters corresponding to the k_i -sized solution. Cluster attributes, such as center and size, are then accessible from the corresponding dendrogram data structure by referencing the clusters in $CurrCl_i$.

<p>Algorithm 1. DENDRO_EXTRACT INPUT: k_i – desired number of clusters i – data partition indicator</p> <pre> 1 begin 2 if $k_i \leq N$ then 3 $CurrCl_i = \{\text{DendrogramRoot}\}$ 4 while $CurrCl_i < k_i$ 5 $MaxCl = \max(CurrCl_i)$ 6 $CurrCl_i = (CurrCl_i / MaxCl) \cup \{MaxCl.Left\} \cup \{MaxCl.Right\}$ 7 return $CurrCl_i$ 8 else Request new k_i 9 end </pre>

DENDRO_EXTRACT starts at the root of the dendrogram 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 3) in the current set of clusters until k clusters are included in the set. $MaxCl$ is the highest element in the current cluster set $CurrCl_i$. 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 complexity, $O(k_i)$, which provides for the real-time extraction of cluster solutions.

Example. Consider a C-TREND data partition D_i whose clustering solution is represented by the dendrogram in Figure 3. Let the desired clustering solution for this partition consist of four

clusters (i.e $k = 4$). To display this solution, the DEDRO_EXTRACT algorithm will start with the root node, cluster 19, and set $CurrCl_i = \{19\}$ (line 3). Since $k = 4$ and $CurrCl_i$ only contains one cluster (line 4), DENDRO_EXTRACT will continue through the while loop (lines 5-6). $MaxCl = \{19\}$ because it is the highest-numbered (and the only) cluster in the $CurrCl_i$ set (line 5), therefore $CurrCl_i$ is updated by replacing cluster 19 by its children nodes clusters 17 and 18 (line 6). In the second iteration, $|CurrCl_i| = 2$ and $MaxCl = \{18\}$, and cluster 18 is replaced by its children, clusters 5 and 14. In the next iteration, $|CurrCl_i| = 3$ and $MaxCl = 17$, which is replaced by its children, clusters 1 and 16. At this point, $|CurrCl_i| = k = 4$ and the algorithm outputs the solution for $k = 4$, which is $CurrCl = \{1, 5, 14, 16\}$.

3.2 Optimal number of clusters

As an initial value k_i for each data partition D_i , C-TREND uses the “optimal” number of clusters based 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, ..., n clusters respectively. C-TREND determines the largest “jump” in MSE across these n solutions, which points to the optimal number of clusters (see Figure 4). 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).

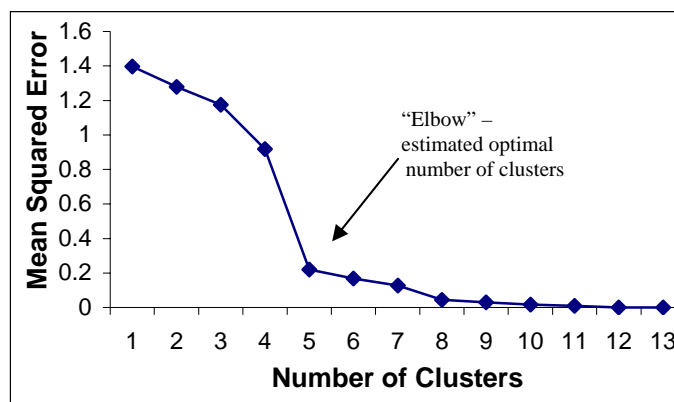


Fig. 4. “Elbow” criterion for optimal cluster number.

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). 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 different k -sized clustering solutions (e.g., for k -means clustering solutions); (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. C-TREND can be extended to support these alternative approaches for determining the optimal number of clusters.

3.3 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 output graphs.

As mentioned above, each data partition possesses a corresponding dendrogram data structure containing all 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 must be 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 a $(2N-1)^2$ possible edges

between two adjacent data partitions. If the data set possesses t data partitions, then there are $(t-1)*(2N-1)^2$ 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 edge list generation would have an 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. 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 one 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 processor with 1 GB of RAM.

Data Partitions (t)	Max Clusters (N)	Num Of Possible Edges	Edge List Creation Time
10	10	3249	0.006 s
100	10	35739	0.062 s
10	100	356409	0.630 s
1000	10	360639	0.650 s
100	100	3920499	6.891 s
10	500	8982009	15.760 s

Table 1. Edge List Creation Times

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 5 contains a plot of the increase in edge list generation time as the number of attributes is being increased from ten to one hundred holding the N and t constant.

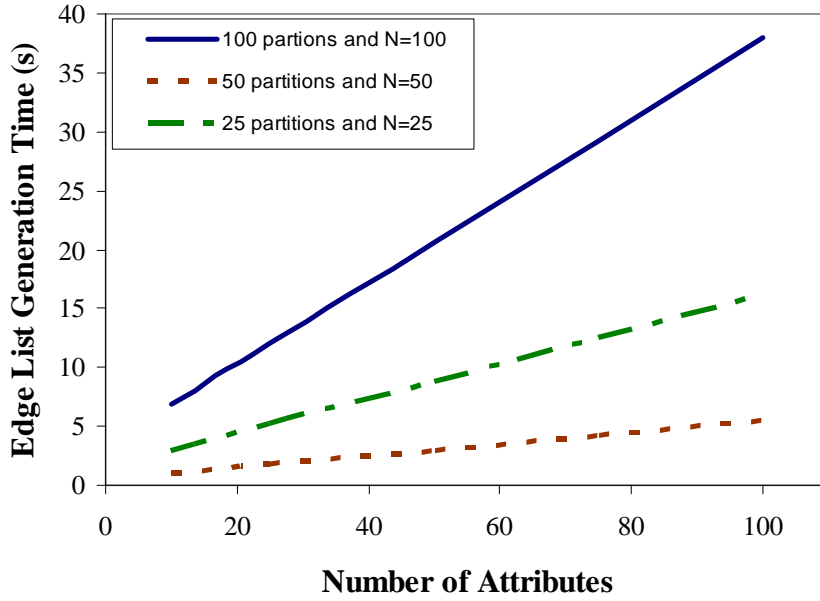


Fig. 5. Time to produce edge list varied by number of attributes.

4. Graph Generation and Interactive Data Visualization

One of the key advantages of C-TREND is its ability to modify the view of the trend graph in real-time based on user input. Domain expertise is of critical importance for the analysis of data mining results, and C-TREND is specifically designed to provide the user with a set of controls for customizing the view of the output trend graphs. Specifically, the user can adjust the *zoom level* of each data partition D_i with the cluster solution size parameters k_i , the *within-period trend strength* with the parameter α , and the *cross-period trend strength* with parameter β .

4.1 Zoom Level

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. Recall that a hierarchical clustering dendrogram is stored for

each data partition, which allows C-TREND to extract different clustering solutions. In particular, 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). The displayed solution is just one of the n possible clustering solutions, and the user may possibly be interested in seeing how the temporal trend changes 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.

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 the underlying clustering solution used to build a trend graph. When the user adjusts k_i , the DENDRO_EXTRACT procedure (Algorithm 1) is rerun and a new clustering solution containing exactly k_i clusters is stored for the i^{th} data partition.

4.2 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 $s = 2$ would likely be spurious for

many practical applications. Additionally, since agglomerative hierarchical clustering starts with n singleton clusters and iteratively combines the two “nearest” clusters, it is possible to have singleton clusters appear in the final k_i -sized cluster solution. To address these issues and provide data visualization control to the user, C-TREND determines if the nodes identified in the initial clustering solution are “strong” enough to be included in the trend analysis based on user input. For every data partition D_i , C-TREND first extracts the clustering solution that contains k_i clusters, and then these clusters are filtered based on a user-defined within-period trend strength parameter α .

C-TREND maintains one α parameter for the entire data set. In other words, each data partition utilizes the same value of α , and by default $\alpha = 0$ (i.e., no filtering). Let V_i be the set of clusters returned from the DENDRO_EXTRACT for data partition D_i , i.e., V_i contains the current 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 in the set of valid nodes U_i that are rendered in the output graph. $\alpha|D_i|$ is the minimum node size threshold, where $|D_i|$ is the number of data points in data partition D_i . $\alpha \in [0,1]$ and represents a percentage of the total number of data points within a data partition. For example, $\alpha = 0.02$ denotes a 2% threshold for the smallest displayed cluster size. A clustering solution for 500 data points with $\alpha = 0.02$ would include only clusters of size 10 or greater. Smaller clusters would be considered spurious and omitted from the solution.

4.3 Cross-Period Trend Strength

Edges are used to represent relationships between nodes (clusters) in adjacent time partitions. Edges are only displayed if the incident nodes are “similar” enough, based on user-defined cross-period trend strength parameter β . Since C-TREND is an interactive data visualization tool, it provides the user the ability to adjust β to include or exclude edges according to their weight

(and, therefore, trends according to their strength) in the output graphs.

An edge is rendered 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 zoom level and within-period trend strength parameters); and (b) the edge weight is less than or equal to the threshold η which depends on β . The edge threshold η is calculated by taking the average of the radii, r , of the two adjacent data partitions, V_i and V_{i+1} , and adjusting it by the user-specified β parameter:

$$\eta = \beta(r(V_i) + r(V_{i+1})) / 2.$$

The radius of a data partition is calculated as the average Euclidean distance from each data point to the center of the data partition. We assign $\beta = 1$ by default, so that all edges stronger than (less than) the average of the two data partition radii are displayed. $\beta \in [0, 1]$ and similar to α , β is a global parameter value for the entire data set.

In our implementation of C-TREND, the distance between two clusters x and y is measured as the Euclidean distance between cluster centers (means) m_x and m_y , i.e., $d(x, y) = \|m_x - m_y\|_2$. C-TREND calculates the cross-period cluster distance $d(x, y)$ for all pair of clusters $x \in V_i, y \in V_{i+1}$ in the adjacent input partitions in the preprocessing phase, and stores this distance as the edge weight in the edge list data structure. In the interactive analysis phase, edges (x, y) that satisfy $d(x, y) \leq \eta$ inequality are rendered in the output graph. 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.

It should be noted that this procedure is easily extensible to support other distance metrics (e.g., Manhattan) and cluster similarity comparison techniques (e.g., average-link metric). Note that if $d(x, y) = 0$ then clusters x and y have identical centers and therefore most likely have a very similar makeup. As an example, let $\beta = 0.8$, $r(V_1)=1.4$, and $r(V_2)=1.8$. Edges will be

generated for cluster pairs (x, y) , where $x \in V_1$ and $y \in V_2$, whenever $d(x, y) \leq 1.28$.

4.4 C-TREND Interface and Output Graphs

C-TREND users have the ability to adjust the parameters k_i ($i = 1, \dots, t$), α , and β , where k_i determines which clustering solution to extract from the dendrogram for partition D_i ; α determines which of the retrieved k_i clusters are strong enough to be included in the output graph for each data partition; and β determines which edges should be included between the clusters from adjacent data partitions. For a given combination of parameter values, C-TREND renders a corresponding output trend graph. Nodes are labeled with the size of the corresponding cluster and edges are labeled with the distance (or similarity) measure for the incident nodes.

Figure 6 provides an example output graph for a data set with 4 data partitions. In this example, the first data partition contains 89 data points, which were divided into four clusters. Using $\alpha = 0.05$, C-TREND filtered out clusters that contained less than 4.45 data points. 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 nodes are displayed 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 η was determined using $\beta = 0.08$ and the average of the adjacent partition radii of the two adjacent partitions. Notice that 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.

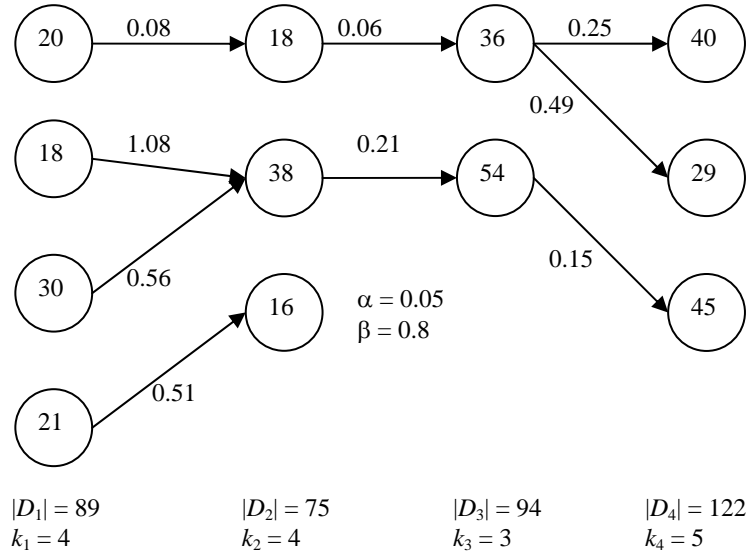


Fig. 6. Example C-TREND output graph.

The C-TREND graphical user interface allows the user to click on each node or edge in the output graph to access additional information. Clicking on a node presents information about the center of the node as a list of data attributes with corresponding values. The radius measure of the node is also displayed. Clicking on an edge provides a list of most significant differences between the two clusters that this edge connects.

4.5 Parameter Implementation

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_i value and the α 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 α is changed, the dendrogram data structure is updated so that only the clusters that are large enough to pass the node filter based on α are assigned a valid α -pass flag. The nodes that have both valid k -pass and α -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 during preprocessing. Each edge in the list possesses a β -pass flag. When β is changed, all edges with a passing weight (based on β) are assigned a valid β -pass flag and all others are assigned an invalid flag. Only edges that have a valid β -pass flag *and* are incident to two valid nodes (nodes with valid k -pass and α -pass flags) are included in the output graph. Using the implementation described above, C-TREND can update output graphs based on user changes to the k_i , β , α parameters very efficiently. Changing any one parameter requires only one operation to update the corresponding flag in the data structure.

5. Performance and Optimization

In this section we discuss the performance of the real-time graph generation process of C-TREND. This process is built on three basic operations: changing k_i (partition clustering solution size) for any of the data partitions; changing α (within-period trend strength) for the entire graph; and changing β (cross-period trend strength) for the entire graph. Each of these operations requires a set of calculations to be performed on the dendrogram data structure, and we will show that, for most practical purposes, these operations are performed efficiently enough to provide “real-time” graph generation and modification.

It should be noted that there is a precedence order for the operations listed above. The k_i value is used to extract all candidate clusters from the dendrogram data structure. If k_i is changed, then C-TREND must reapply the node and edge filters to generate the updated trend graph, even if α and β have not changed. Additionally, the α value is used to determine which of the candidate clusters will be included in the graph as nodes. If α is changed, then the set of nodes for each partition may have changed and therefore the edge filter must be reapplied. By implementing the k -pass, α -pass, and β -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.

Theoretical efficiencies of the parameter changing operations can be easily computed based on the maximal number of clusters N and the number of data partitions t . Changing α requires C-TREND to scan through all possible nodes to determine if each node should have a valid or invalid α -pass flag. If each partition contains $2N-1$ possible nodes and there are t partitions, changing α has an asymptotic upper bound of $O(Nt)$. Similarly, β is also a graph-wide parameter, and changing β requires C-TREND to scan through all possible edges to determine if each edge should possess a valid or invalid β -pass flag. Since there are $(t-1)(2N-1)^2$ possible edges, changing β 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 until it has found the k_i clusters that make up the k_i -sized clustering solution. In fact, DENDRO_EXTRACT performs one operation for

each of k_i clusters in the solution and therefore DENDRO_EXTRACT is $O(k_i)$. Changing k_i simply uses DENDRO_EXTRACT to update the k -pass flags of the dendrogram data structure and because each partition has at most $2N-1$ clusters, changing k_i would require at most $2N-1$ operations. Therefore changing k_i is linear in the maximum sized clustering solution or $O(N)$.

To optimize changing k_i even further C-TREND uses two very 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*: $CurrCl_{k+1} = (CurrCl_k / \{MaxCl\}) \cup \{MaxCl.Left\} \cup \{MaxCl.Right\}$. To decrease the number of clusters included in the partition by one (i.e., $k := k-1$), let v_1 be the root node and v_{k-1} be the node at a height of $k-2$ from the root node in the dendrogram. For example, referring to Figure 3, v_3 would be node 17 and v_6 would be node 14. By construction, the k cluster solution will contain the children of v_{k-1} . Therefore, the $k-1$ clustering solution is derived by replacing the children of v_{k-1} with the node v_{k-1} . In other words, $CurrCl_{k-1} = CurrCl_k / \{v_{k-1}.Left\} / \{v_{k-1}.Right\} + \{v_{k-1}\}$, where $v_{k-1}.Left$ and $v_{k-1}.Right$ are the left and right children of node v_{k-1} . This optimization reduces the changing k_i operation to constant time $O(1)$ for increments and decrements of k_i .

Data Partitions (t)	Max Clusters (N)	Change k_i	Incr. k_i	Decr. k_i	Change α	Change β
10	10	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s
10	100	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	0.007 s
10	500	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	0.192 s
100	10	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	0.001 s
1000	10	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	0.009 s
10	10	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s
10	10	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s
100	100	< 1 μ s	< 1 μ s	< 1 μ s	< 1 μ s	0.084 s

Table 2. Experimental analysis of C-TREND real-time performance

To validate the 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 eight data sets ranging in the number of data partitions (t), the maximum-sized cluster solution (N) for each partition, and the number of attributes per data point. For each experiment we ran the operation for five different randomly generated data sets and Table 2 displays the average time to perform each partition. Notice that the changing k_i and incrementing/decrementing k_i operations took less than 1 μ s regardless of the size of the data set used. Additionally, changing α also took less than 1 μ s for each dataset analyzed. Only the changing β operation required processing times greater than 1 μ s. For the data set with a maximum sized clustering solution of 500, changing β took 0.192 s. Recall that the changing β operation has the computational complexity of $O(N^2t)$, and that for most practical purposes it is sufficient to have $N \leq 50$. This suggests that changing β will typically take less than 0.084 s (the last row in Table 2) for typical uses of C-TREND. Additionally, the increase in processing time for these operations from 0.007 s to 0.192 s corresponding to the increase of $N = 100$ to $N = 500$ agrees with our theoretical calculations of $O(N^2t)$ since $0.007 * 25 = 0.175$ (which is very close to 0.192). The experimental results in Table 2 support our claims that C-TREND output graph generation and modification can be done in “real-time” and provides 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.

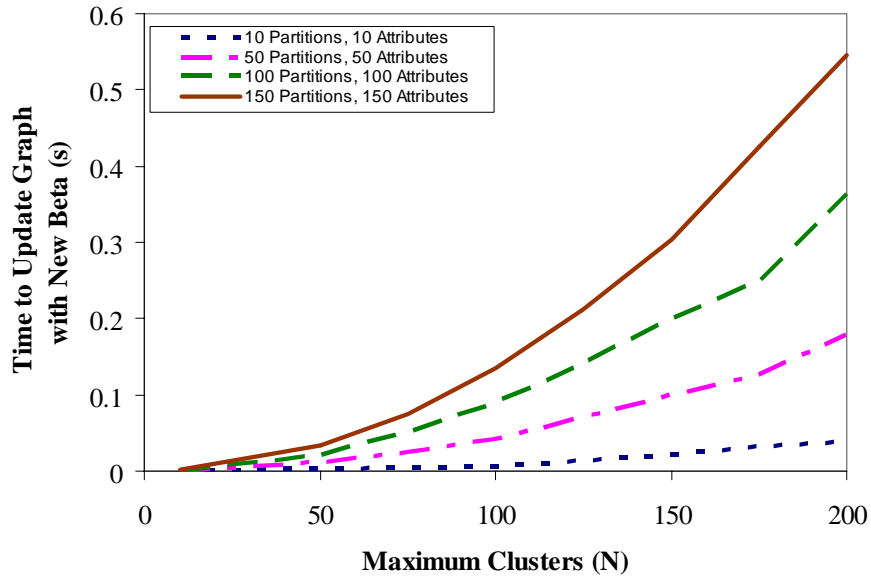


Fig. 7. Time to update graph with new β value (with varying N)

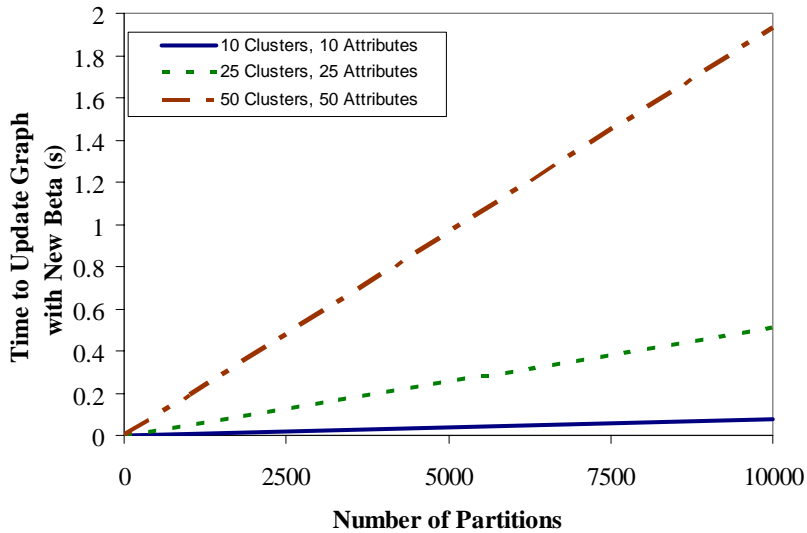


Fig. 8. Time to update graph with new β value (with varying number of partitions t)

To demonstrate the scalability of the C-TREND technique, we report a second set of experimental results in Figures 7 and 8. We demonstrate the efficiency of C-TREND in its most computationally costly operation – updating β – for various large data sets. In Figure 7, we show that, while increasing the maximum sized clustering solution N , C-TREND can still update β relatively quickly. In fact, for a data set with 150 data partitions, $N = 200$, and 100 attributes per

data point, β is updated in less than 0.6 seconds. Figure 8 demonstrates that dramatically increasing the number of data partitions can also be handled by C-TREND in a real-time fashion. In fact, for the extremely large data set containing 10,000 partitions with $N=50$ and 50 attributes per data point (approximately 24,000,000 edges), β is updated in less than 2 seconds.

6. Wi-Fi Technology Case Study

C-TREND is a versatile technique for identifying and representing trends in temporal multi-attribute data. As mentioned earlier, the user can vary three important parameters for determining the output trend graphs: partition zoom level k_i , within-period trend strength α , and cross-period trend strength β . In addition, C-TREND is highly customizable and can incorporate multiple clustering methods and distance metrics. To demonstrate the use of C-TREND for identifying trends in transactional attribute-value data and show how modifying input parameters affects the trend graph output, we present the analysis of data on over 2,400 certifications for new wireless networking (802.11) technologies awarded by the Wi-Fi Alliance (wi-fi.org).¹

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 set of standards including IEEE protocol (802.11a, b, g, d, and h), security protocol (e.g., WPA and WPA2), authentication protocol (e.g., EAP and PEAP), and quality of service (e.g., WMM). Each product certification consists of a date of certification and a set of binary attributes indicating the presence of the standards listed above.

For our analysis, the certification data included all standards-related attributes as well as

¹ 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.”

the product type (e.g., PDA, internal card) and product category (component, device, and infrastructure) attributes. Certifications were coded into product categories based on the similarity of their product type and functionality; e.g., 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 existing products, such as PCs or laptops.

Figures 9a and 9b 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 9a 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, 0.15, 0, 0, 1, 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 it 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% internal cards, and 15% USB client devices. Of these components, 100% are 802.11b-certified and 4% have WPA-personal, WPA-enterprise, and EAPTLS certifications.² 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 in clusters over time. For example, the edge labeled 0.08 in Figure 9a 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, which indicates the clusters are extremely similar. Following this same trend into 2002, we see an edge with weight 0.04. This indicates that during 2000-2002 years 802.11b access points with very similar technical specifications was a dominant technology type in the set of all

² 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).

Comparing Figures 9a and 9b we show the effect of modifying the α and β parameters. Figure 9a presents a C-TREND trend graph using $\alpha = 0.02$ and $\beta = 0.8$, which excludes clusters smaller than 2% of the total number of data points in a data partition and edges less than 80% of the average period radius. Figure 9b has more relaxed parameters, the default $\alpha = 0.0$ and $\beta = 1.0$ and, therefore, includes additional nodes and edges (indicated by dashed lines). 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, in Figure 9b 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 in Figure 9a. On the other hand, Figure 9b includes many clusters of size 1 with no adjoining edges. These are likely spurious events that do not provide insights on trends and can be filtered out of the graph using more restrictive parameters, as in Figure 9a.

In addition to adjusting the α and β parameters, the user can also vary the clustering output with the k_i parameter. Figure 10 presents a hierarchical clustering solution for periods 2003, 2004, and 2005, which allowed the user to “zoom out,” i.e., to display a more aggregated view of the data by automatically combining the most similar clusters. Specifically, we used $k_i = 2$ for periods 2003 and 2005 and $k_i = 3$ for 2004. Compared to Figure 9, adjusted k_i values provide the user with a means for aggregating data into *fewer* clusters and, therefore, to identify higher-level trends. Note that the α parameter has not caused any additional filtering. Adjusting the clustering criteria results in a more streamlined graph that identifies two major trends from 2000 to 2002 (i.e., 802.11b access points and 802.11b network cards) and then two distinct new trends (i.e., the 802.11b/g WPA security certified products and the internal cards with all standards).

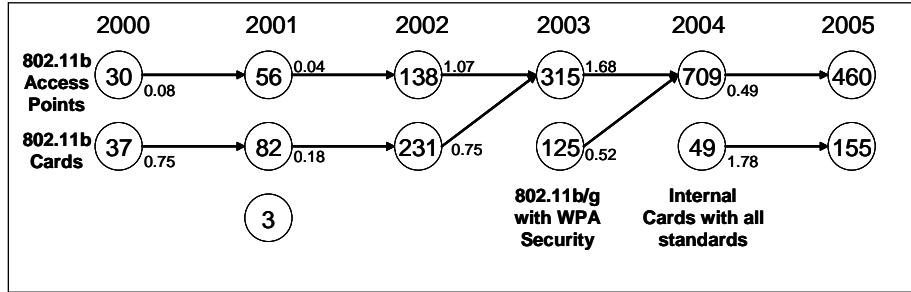


Fig. 10. Adjusting k_i for a “zoom-out” view with $\alpha = 0.02$, $\beta = 0.8$

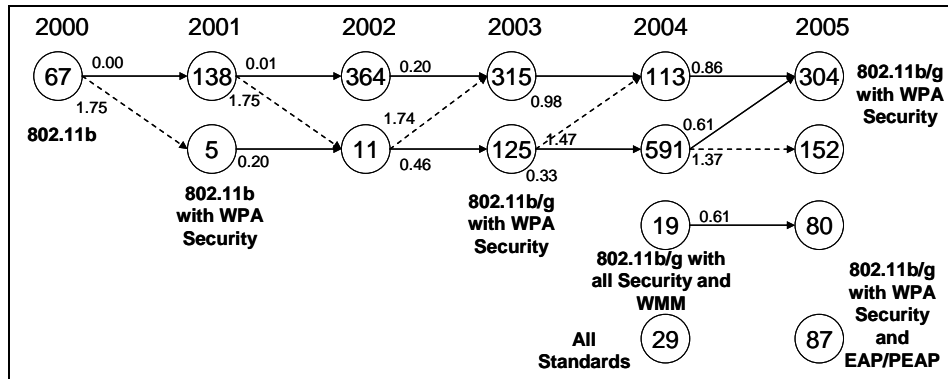


Fig. 11. Wi-Fi trend graphs with $\alpha = 0.02$, $\beta = 1.0$, and “standards-only”

In addition to the three trend parameters, C-TREND can also be used to plot output graphs for various attribute sets and partition schemes for the same data set. Figure 11 presents a C-TREND graph of the Wi-Fi alliance certification data with product type and product category attributes removed, focusing only on standards-related data attributes. The clustering criteria for this graph are the same as those presented in Figures 9a and 9b. In this example, the absence of product-related attributes results in the identification of clusters based solely on the certification standards and two clear trends are identified: products *with* and *without* security certification. In 2004 it is also apparent that the WMM quality-of-service standard begins to arise. In contrast, a clear start to WMM certification was not apparent in Figures 9a and 9b, because it was masked by the multiple clusters based on product type.

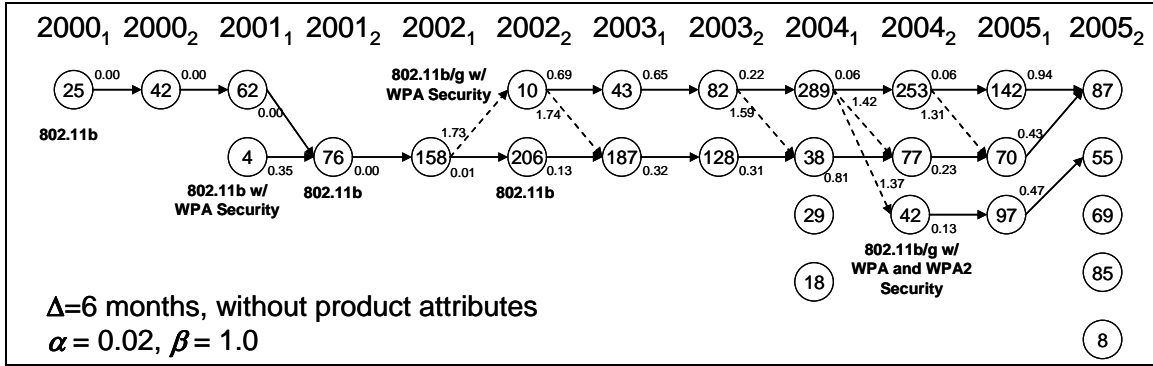


Fig. 12. Wi-Fi trend graph for six month periods: “standards-only” view.

Finally, Figure 12 depicts a trend graph using a different time period size, where the data set used for Figure 11 is partitioned into *6 month* intervals. The smaller time window provides a finer-granularity representation of the temporal trends and doubles the number of data partitions. Compared to Figure 11, Figure 12 clearly shows that the first products with WPA security appear in early 2001, but then seem to disappear in late 2001 and do not reappear until late 2002. It also shows some additional trends in the late 2004 and 2005 time periods. These “finer” trends were not visible in Figure 11 due to the larger time period partition.

7. Discussion and Conclusion

By harnessing computational techniques of data mining, we have developed C-TREND, a novel method to uncover, analyze, and visualize trends in multi-attribute temporal data. The proposed technique is versatile and customizable 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 and adjust the levels of granularity. C-TREND is also extremely scalable. As demonstrated in Section 5, the time required to adjust trend parameters is quite low even for larger data sets, which provides for real-time visualization capabilities. Furthermore, C-TREND is applicable in many different temporal data analysis contexts; for instance, C-TREND graphs

can aid firms performing historical analyses or making decisions for future directions based on historical trends.

This work provides many directions for future research. C-TREND graphs provide a general framework for developing new trend analysis techniques. We plan to extend C-TREND by developing a 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. Furthermore, C-TREND provides the initial mechanics for developing predictive models and hypotheses for the existence, birth, death, and continuation of trends in data.

References

- M.S. Aldenderfer and R.K. Blashfield, *Cluster Analysis*, Sage Publications, Newbury Park, CA, 1984.
- D. Arthur, S. Vassilvitskii, "How Slow is the k-means Method?," *Proceedings of the 2006 Symposium on Computational Geometry (SoCG)*, 2006.
- P. Brockwell, and R. Davis, *Time Series: Theory and Methods*, Springer-Verlag, New York, 2001.
- R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd Edition, Wiley-Interscience, New York, 2000.
- A. Jain, M. Murty, and P. Flynn, "Data Clustering: A Review," *ACM Comp. Surveys*, vol. 31, no. 3, pp. 264-323, 1999.
- N. Jardine and Sibson, R. "The construction of hierarchic and non-hierarchic classifications," *The Computer Journal*, vol. 11, no. 2, pp. 177-184, 1968.
- S. C. Johnson, "Hierarchical Clustering Schemes," *Psychometrika*, vol. 32, no. 3, pp. 241-254, 1967.
- S. Y. Jung and T.-S. Kim, "An Agglomerative Hierarchical Clustering Using Partial Maximum Array and Incremental Similarity Computation Method," *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp. 265-272, 2001.
- L. Kaufman, and P. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, Wiley, 1990.
- E. Keogh, S. Kasetty, "On the need for time series data mining benchmarks: A Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349-371, 2003.
- G.W. Milligan and M.C. Cooper, "An examination of procedures for determining the number of

- clusters in a data set,” *Psychometrika*, vol. 50, no. 2, pp 159—179, 1985.
- J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, Mei-Chun Hsu, “Mining sequential patterns by pattern-growth: The PrefixSpan approach,” *IEEE Trans. Knowledge and Data Engineering*, vol. 16, no. 10, pp.1-17, 2004.
- J. Roddick and M. Spiliopoulou, “A survey of temporal knowledge discovery paradigms and methods,” *IEEE Trans. Knowledge and Data Engineering*, vol. 14, no. 4, pp.750-767, 2002.
- C.A. Sugar, “An application of cluster analysis to health services research: Empirically defined health states for depression from the sf-12,” *Stanford University, Department of Statistics Technical Report*, 1999.
- C.A. Sugar and G.M. James, “Finding the number of clusters in a data set: An information theoretic approach,” *Journal of American Statistical Association*, vol. 98, pp 750-763, 2003.
- R. Tibshirani, G. Walther, and T. Hastie, “Estimating the number of clusters in a data set via the gap statistic,” *Journal of the Royal Statistical Society*, vol. 63, no. 2, pp. 411-423, 2001.
- M. Zaki, “SPADE: An efficient algorithm for mining frequent sequences,” *Machine Learning*, vol. 42, no. 1-2, pp. 31-60, 2001.