RC 21847 (98328) 2 October 2000 **Computer Science**

IBM Research Report

A Systematic Approach to Discovering Correlation Rules for Event Management

L. Burns, J. L. Hellerstein, S. Ma, C. S. Perng, D. A. Rabenhorst
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

D. Taylor
University of Waterloo
Ontario, Canada

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

A Systematic Approach to Discovering Correlation Rules In Event Management

L. Burns, J.L. Hellerstein, S. Ma, C.S. Perng, D.A. Rabenhorst
IBM T.J. Watson Research Center
Hawthorne, New York

D. Taylor
University of Waterloo
Ontario, Canada

Abstract

For large installations, event management is critical to ensuring service quality by responding rapidly to exceptional situations. Key to this is the use of correlation rules that relate situations--in terms of event patterns--to actions. However, a central difficulty remains--determining what rules to write. Herein, we propose a way to reduce this burden by visualizing and mining event histories to discover patterns in event data. Our experience with a wide variety of production data has identified several patterns of interest (e.g., event bursts and partial periodicities). We apply our methodology to these data. In addition, we describe a tool we have developed to aid in pattern discovery.

1. Introduction

Event management is central to the operations of computer and communications systems in that it provides a way to focus on exceptional situations. As installations grow in size and complexity, event management has become increasingly burdensome. Automated operations (AO), which was introduced in the mid 1980s (e.g., [Mill86]), provides a way to automatically filter and act on events so as to detect problems, diagnose their root cause, and take corrective action. However, doing so requires developing correlation rules, a knowledge-intensive and time-consuming task. Herein, we propose a way to reduce the burden of developing correlation rules by visualizing and mining event histories to discover patterns in event data.

The top part of Figure 1 summarizes how event management is done today. Events flow into the event management system, where they are parsed and stored. Then, a rule matching or correlation engine uses (correlation) rules to interpret these events, some of which invoke actions (e.g., alarms, emails).

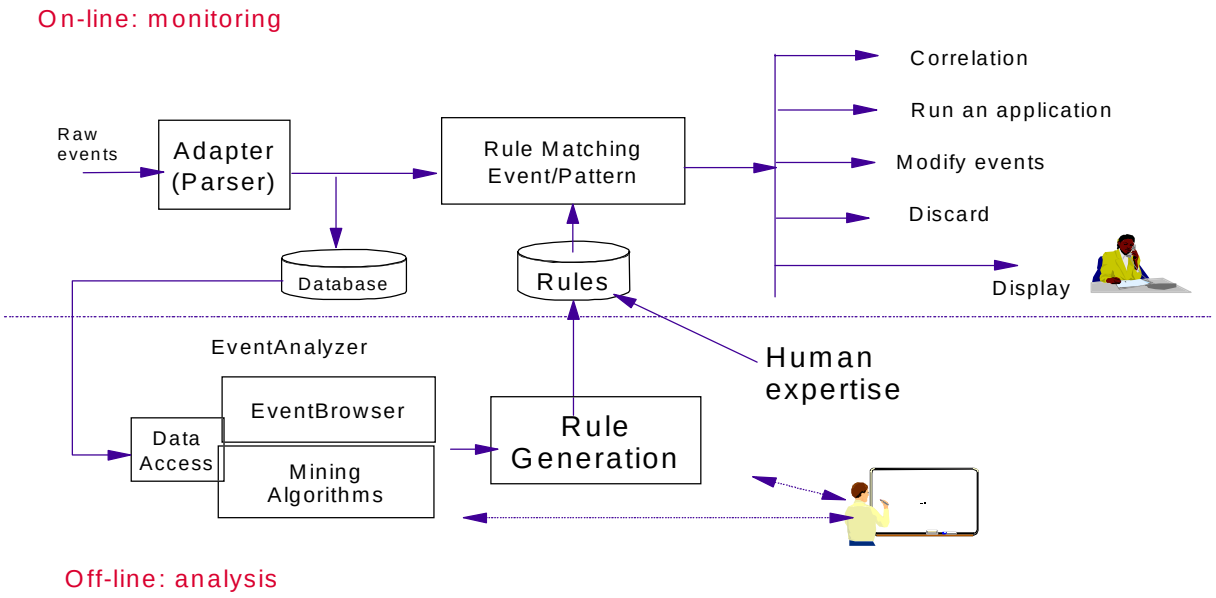

Figure 1: Architecture for Integrated On-Line and Off-Line Analysis

Correlation rules are structured as if-then statements. The if-part (or left-hand side) describes a situation in which actions are to be taken. The then-part (or right-hand side) details what is to be done when the condition is satisfied. Our definition of a correlation rule is quite broad in that it encompasses any way to specify the detection of a situation in which actions are taken (e.g., as in [YeSkEyYeOh96]).

Constructing correlation rules requires considerable effort. Our focus is constructing the left-hand side. Today, two broad approaches are : a model-based approach and a knowledge-based approach. The model-based approach employs a dependency model, such as topology information to make inferences about connectivity problems (e.g.,[YeSkEyYeOh96]). If such information is available and can be maintained in a cost-effective way, there is tremendous advantage in its exploitation. Unfortunately, for many problems (e.g., software errors), this is not the case. The knowledge-based approach uses domain experts to construct left-hand sides, a time-consuming and expensive undertaking. Three factors contribute to this. First, domain experts are scarce in virtually all installations. Second, in large installations, a group of experts is often required (since no one person has the requisite knowledge), but it is often difficult to synthesize an answer from the opinions of many experts. Finally, even if knowledge can be acquired, knowledge maintenance is problematic due to the dynamics of network systems. We note that both approaches require experts to supply knowledge. The difference is when this knowledge is supplied. In the first case, it is supplied during development of the management system. In the second case, it is supplied during installation customization. Our focus is on reducing the cost and the time of the both approaches. The intent is to go well beyond tools that simplify the mechanics of rule construction, such as visual programming techniques. Rather, we seek ways to aid in determining the *content* of the rules themselves.

How can this be done? Our key observation is that almost all installations maintain repositories of event logs for post-mortem analysis. These rich historical data provide valuable information for situations in which correlation rules should be developed. Our broad approach is to visualize and mine these logs to discover correlation rules for event Data mining is a mixture of statistical and data management techniques that provide a way to cluster categorical data so as to find "interesting" combinations (e.g., hub "cold start" trap is often preceded by "CPU threshold violated"). Considerable work has been done in mining transactional data (e.g., supermarket purchases), much of which is based on [AgImSw93] and [AgSr94]. For event data, time plays an important role. Follow-on research has pursued two directions that address this requirement. The first, sequential data mining (e.g., [AgSr95]), takes into account the sequences of events rather than just their occurrence. The second, temporal data mining (e.g., [MaToVe97]), considers the time between event occurrences. Data mining has been applied to numerous domains. [ApHo94] discuss the construction of rules based for capital markets. [CoSrMo97] describes approaches to finding patterns in web accesses. [ApWeGr93] discusses prediction of defects in disk drives. [MaToVe97] addresses sequential mining in the context of telecommunications events. The latter, while closely related to our interests, uses event data to motivate temporal associations, not to identify characteristic patterns and their interpretation. In summary, while much foundational work has been done on data mining and some considerations have been made that pertain to mining event data, these efforts have addressed the specifics of patterns that arise in enterprise event management nor have they considered the characteristics of event data for systems management.

What are these specifics and characteristics? First, the temporal information of events is very important, but it is seldom accurate. This requires techniques that have some tolerance for skewed timestamps. Second, event management focuses on problematic situations, which should be *infrequent* (or else there will be personnel changes). Infrequent patterns, which are difficult and expensive to find, have not been the focus of existing data mining research. Third, we are typically dealing with a large amount of data. We have seen thousands of events per day for a midsize installation or even millions of events per day in a large, complex installation. Usually, it is necessary to explore several months worth of data. This requires scalable tools [TaHaHeMa2000].

We are aware of one effort that relates closely to our own. [KlMaTo99] describe a system for visualizing and mining telecommunications alarm data for the purpose of constructing correlation rules. While the goal is similar, there are importance differences in the specifics. First, their focus is on frequently occurring patterns. As just noted, infrequent but highly likely associations are actually of much more interest. Second, while they propose an approach that is interactive, the iterative and systematic examination of patterns is not considered. Third, while both of us see value in combining visualization and mining, we emphasize different aspects. [KlMaTo99] explores visualization to aid in interpreting the results of mining. We focus on mining to aid visualization, such as the algorithms incorporated into our tool that order categorical data to improve visual pattern recognition [MaHe99b]..

Herein, we develop a system that integrates visualization and data mining for discovering actionable patterns in event logs. Our work can be best described by the lower part of Figure 1. Event logs are stored in either flat files or a database. The data access component provides an interactive way for a user to select a subset of data for analysis through constraints. Then, visualization techniques and mining algorithms are applied to the data selected to identify patterns that should be translated into correlation rules.

This paper proposes a methodology for pattern discovery, and a system that supports this methodology. A preliminary version of this system, which we refer to as **EventBrowser**, was discussed in [MaHe99a]. EventBrowser uses visualization and data manipulation capabilities provided by Diamond/GEM [Ra94] to summarize and link event data, allowing for navigation at different resolutions and exploring the temporal and spatial relationships between events. These capabilities are augmented by algorithms for mining event data to find several kinds of patterns: event bursts, periodic event patterns, and similarity patterns. EventBrowser and our mining algorithms have been used to analyze a variety of operational event data including logs from: web servers, Netview, Tivoli Event Console, and Dynamic Host Configuration Protocol servers. Numerous actionable patterns have been discovered, including: those related to network problems, hub mis-configurations, and spurious monitoring.

The remainder of this paper is structured as follows. Section 2 describes a methodology for systematically discovering correlation rules from event histories. Section 3 presents several examples that incorporate this methodology, including visualizations and data mining techniques that aid the discovery process. Section 4 details the structure of the EventBrowser, especially how its graphical user interface is componetized. Our conclusions are contained in Section 5.

2. Overall System and Methodology

Typically, system administrators know that a given problem scenario *might happen* or, worse yet, they might just *wait until it happens* to take any action. Thus, we hope that through our discovery-based approach it will be possible to use patterns of events to identify problems before they lead to widespread service disruptions. In this section, we describe our methodology for discovering correlation rules from historical event data. This methodology exploits the visualization and mining capabilities of the bottom part of Figure 1.

Figure 2 illustrates the methodology that we propose. We describe each step and the role that EventBrowser plays in that step.

1. Patterns are identified. An example of a pattern is a sequence of port-up and port-down events. EventBrowser supports both visual exploration and data mining for discovering patterns.

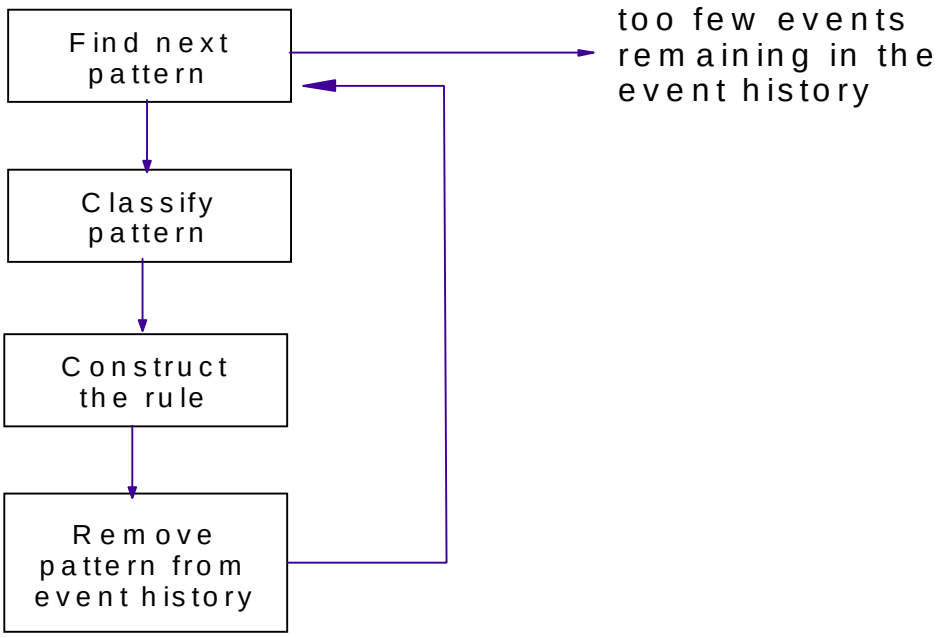

Figure 2: Methodology for Discovering Correlation Rules

2. The identified patterns are classified by domain experts. This is assisted by EventBrowser through its capability to find similar patterns. Two classifications are considered. If the pattern is known or not actionable, it is classified as "uninteresting". On the other hand, if the pattern has not been encountered previously and is actionable, it is considered of interest. In the latter case, an action may be associated with the pattern. A correlation rule is constructed. The left-hand side of the rule describes the pattern employed for on-line matching purposes. An example here is "If there is a port-up event for an IP address followed by a port-down event for the same IP address". The right-hand side encodes the action to take if the pattern is encountered. Uninteresting patterns are removed from the data. Interesting patterns either generate an alarm or take an action specified in step 2.

3. The constructed rule is used to remove the patterns from the event histories employed in the discovery process. This allows unknown and more subtle patterns to be found in the next iteration of steps 1-4. EventBrowser supports this step by providing ways to remove events in a pattern.

Note that this methodology is both interactive and iterative. The spirit here is very much in line with statistical process control (e.g., [BaNi93]). A process is modeled by using an algebraic function of a set of independent variables to changes in dependent variables under both normal and abnormal conditions. A good model is one that explains as much of the change (usually quantified by variability) as possible.

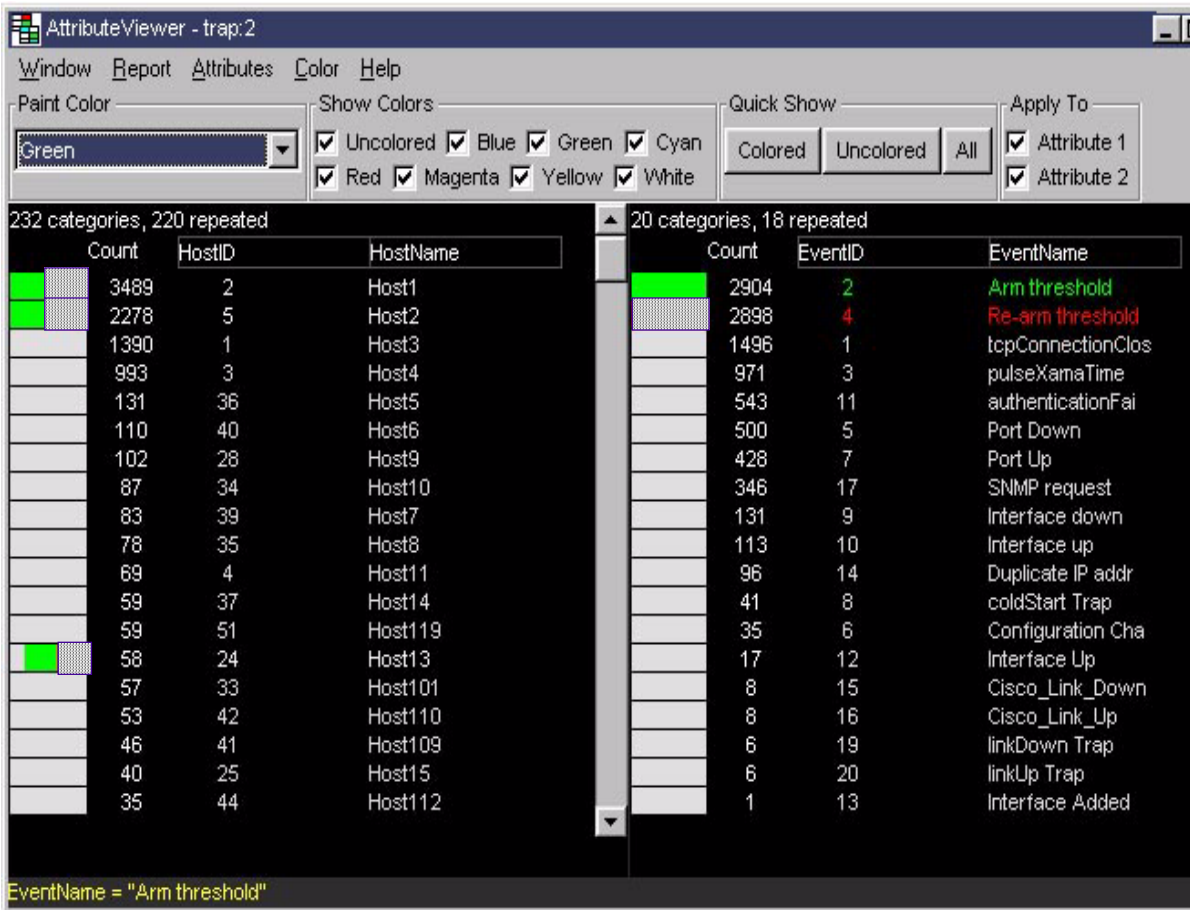

Figure 3: Summary Tables Used in Pattern Discovery

Our analogy to the algebraic model is a pattern. We seek a set of patterns that covers as many of the events as possible. Since it is rare that this can be accomplished by using a single pattern, we must have a way to systematically identify a set of patterns that achieve this objective. Our approach is to incrementally (a) find the pattern that explains what is dominant and (b) remove this from the event history so that more subtly patterns can be discovered. We note in passing that such considerations are not addressed in [KlMaTo99].

3. Examples of Pattern Discovery

To illustrate our methodology, we will present examples of pattern discovery using visualizations of event data and pattern-finding algorithms. Each example describes how the left-hand side of

rules can be constructed from these patterns These examples and the supporting visualizations motivate the discussion of EventBrowser in Section 4.

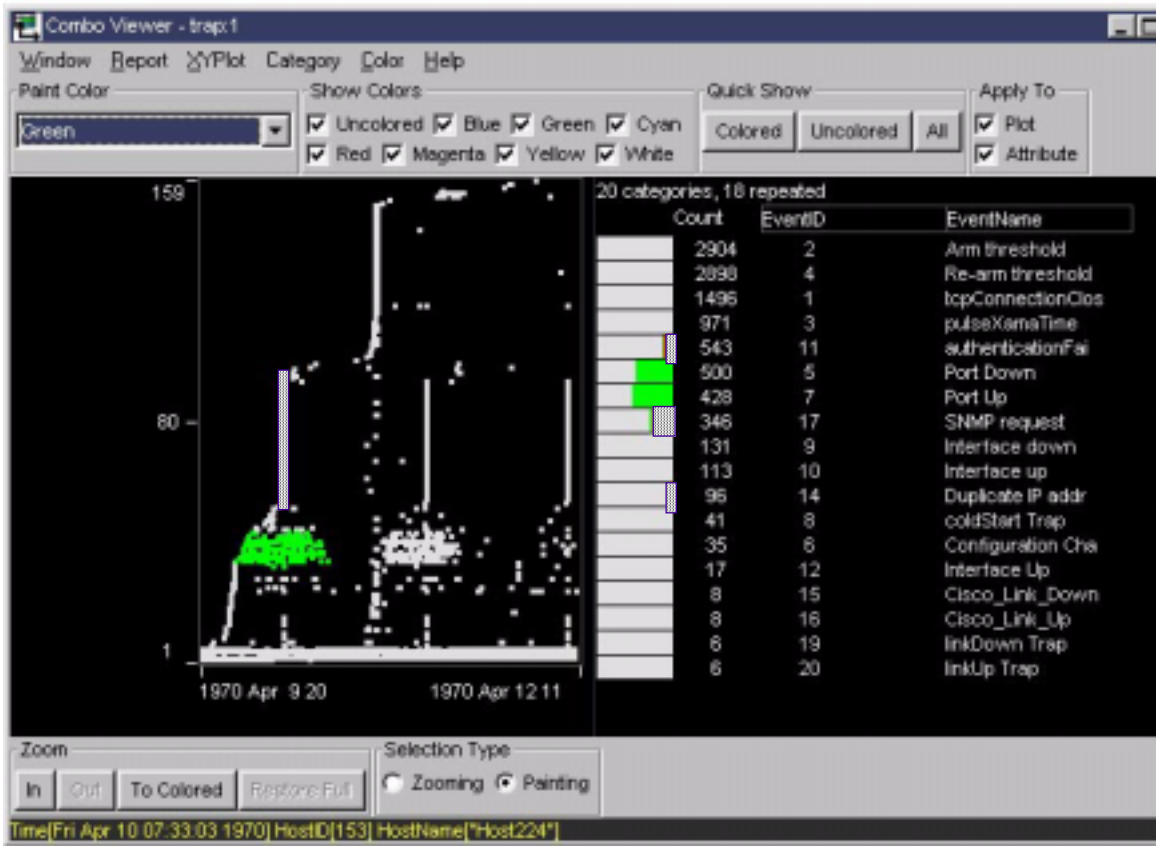

Figure 4: Combining Summary and Temporal Information

3.1 Frequent Events

An intuitive starting point for discovery is to examine frequently occurring events. We consider summary information for SNMP trap data in a corporate intranet. For each event, we have extracted the time at which the trap was received by the management station, the host from which the event originated, and the event type. The latter two are categorical data in that they have no inherent numerical order (although an order can be assigned).

Figure 3 displays two summary tables for the data used in this example (with host names changed to sanitize the data). The left table lists hosts in descending order by the number of events that originate from that host. The right table does the same for event types. Color is used to link the tables, although in this case we have used a fill pattern. Note that hosts 1, 2, and 13 account for

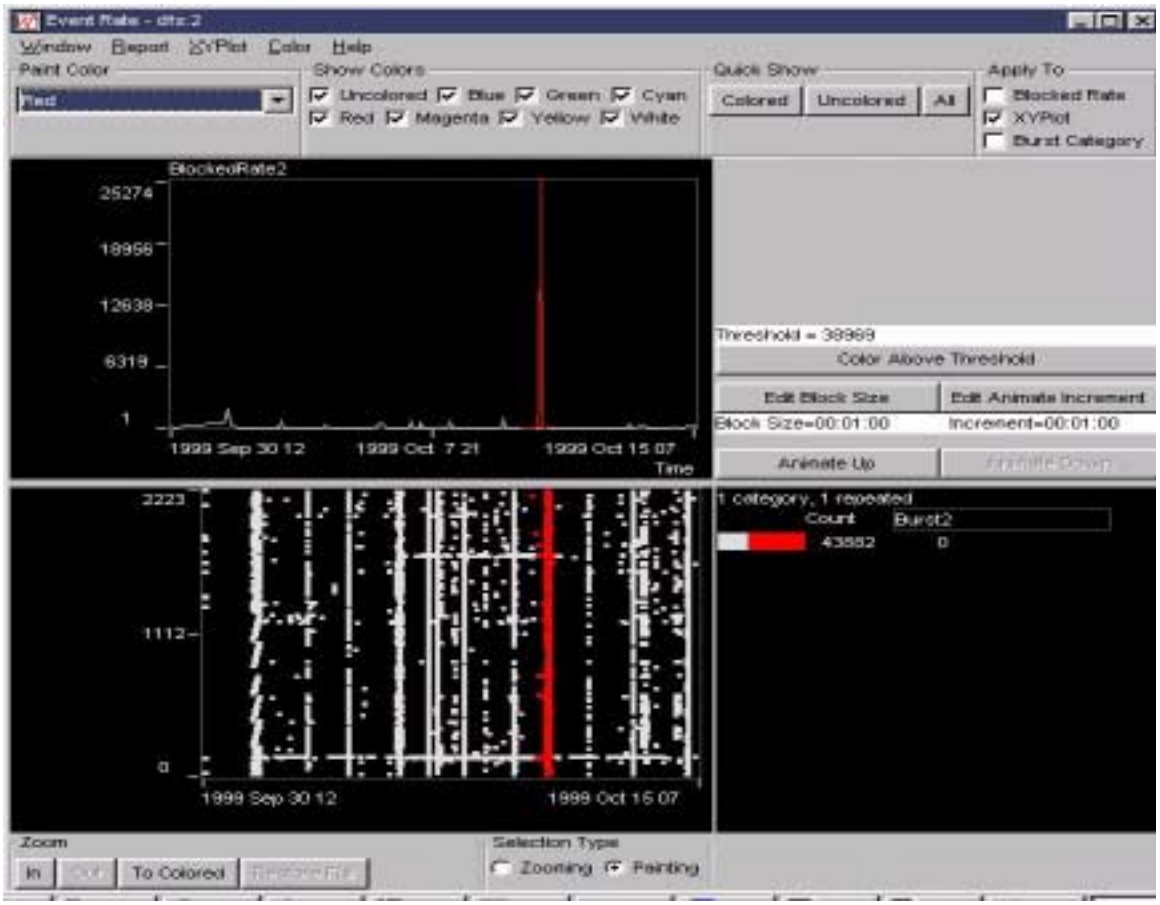

Figure 5: Event Burst Pattern

the vast majority of the events in this network and that most events are either “arm threshold” (threshold violated) or “re-arm-threshold” (threshold no longer violated). Further, we see that the number of “arm threshold” and “re-arm threshold” events is approximately balanced on hosts 1, 2 and 13.

The pattern shown in Figure 3 is normal in that, while the thresholds were violated frequently on these hosts, it was not considered exceptional for hosts 1 and 2. In this case, we would like to filter out this pattern from our data so we can go on to explore more interesting (or troublesome) events. A simple approach to constructing a rule left-hand side is to specify events based on their type and host of origin

(type=”Arm threshold” OR type =”Re-arm threshold”)

AND (host = host1 OR host=host2 OR host=host13)

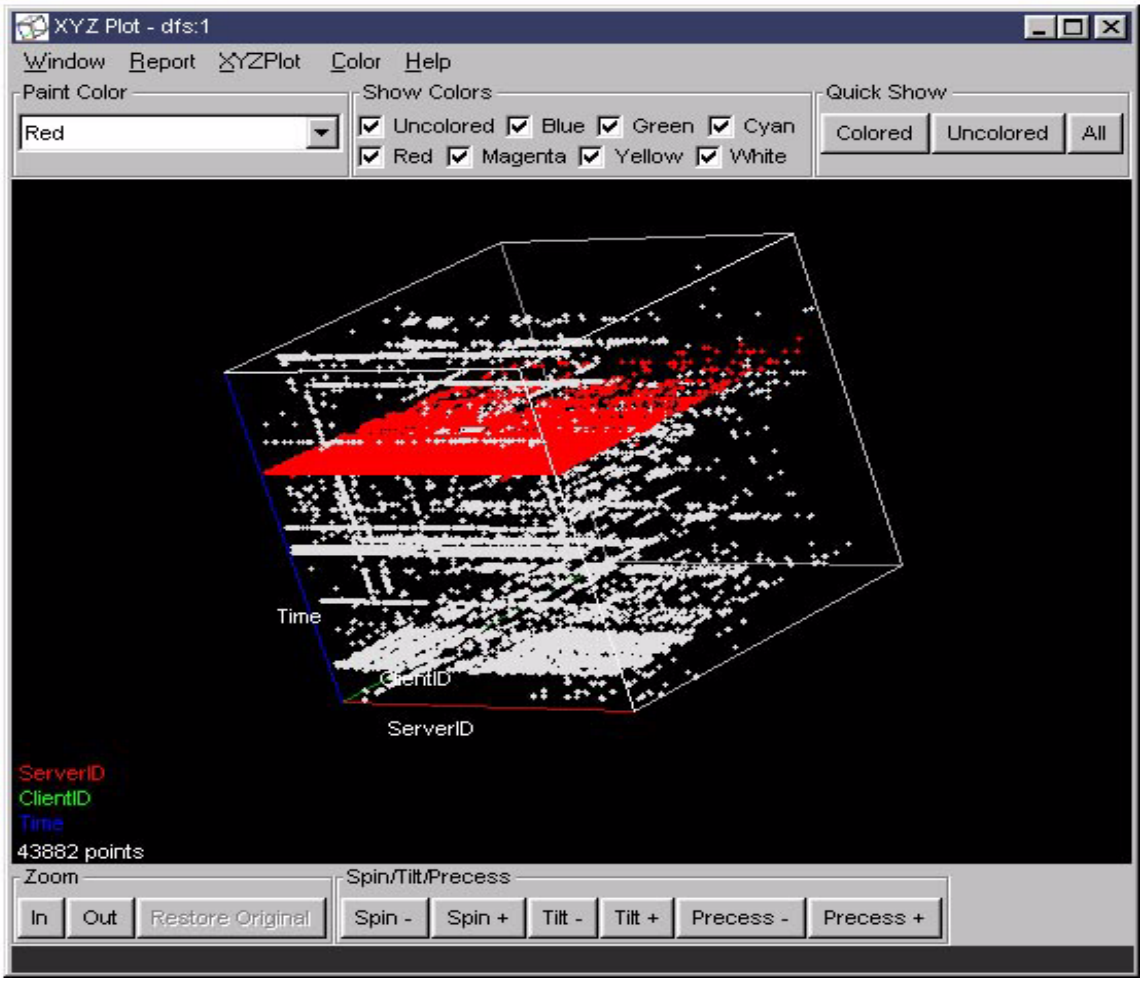

Figure 6: Three Dimensional Plot of "lost contact" Events

We view this as a kind of query (much like a where-clause in a SQL statement). In this example, the events are filtered. Thus, the full rule is:

If (type="Arm threshold" OR type ="Re-arm threshold")

AND (host = host1 OR host=host2 OR host=host13)

Then filter these events

3.2 Frequent Events With Temporal Information

While summary information is extremely useful, sometimes we need to localize events in time as well. To be specific, we know that the "Arm threshold" and "Re-arm threshold" in Figure 3 are generated primarily by hosts 1, 2, and 13. However, we know nothing about when they are generated or how closely in time they occur. Figure 4 shows a scatter plot (left-hand side) in

combination with a summary table (right-hand side). The temporal information present in this view allows us to identify patterns that are not apparent in the summary of categorical data in Figure 3.

Specifically, many patterns are apparent when temporal information is included. There is a "spike" pattern, which is indicated by the dotted fill pattern, and a "cloud" pattern. The events making up the "spike" are "SNMP request" and "authentication failure". Most likely this is a monitor that was not deactivated. However, it can be indicative of a port scan, which is a leading indicator of security intrusion. This is definitely a pattern of interest to the operations staff. One way to describe this in the left-hand side of a rule is:

(type="SNMP request" OR type ="authentication failure")

AND rate > 10/second

AND number of hosts > 10.

The rule's right-hand side will invoke an action, at least sending an alert to an operator.

The "cloud" pattern is made up of "port up" and "port down" events. The hosts from which events originate are hubs. This is a normal pattern resulting from the fact that several times

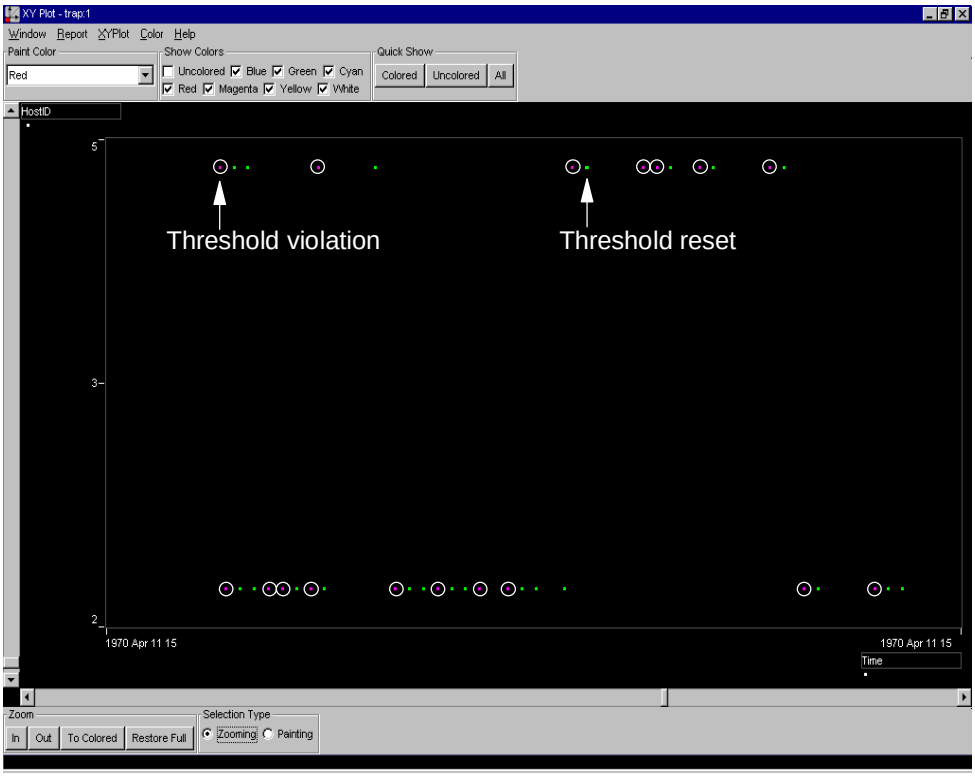

Figure 7: Periodicity Related to Threshold Violations

throughout the day, users plug and unplug their laptops in the subnet. Here, the left-hand side might be

(type="port up" OR type ="port down")

AND host is a hub

This is a normal pattern and should be filtered so operations do not see it.

3.3 Event Bursts

A common situation in practice is for a single cause (e.g., a router failure) to have widespread impact. This is reflected as an event storm or an event burst. To illustrate, we consider a different data set, also from a corporate intranet. These events are all of the same type--lost contact between a client and a server. Thus, the event attributes considered here are: client, server, and time.

Consider Figure 5, which contains two weeks of lost contact events. The figure is divided into four quadrants. The bottom left-hand quadrant is a scatter plot in which time is the x-axis, and the server generating the lost contact is the y-axis. (Servers are ordered to facilitate visualization.). The upper left quadrant indicates the rate of events corresponding to the scatter plot below. These rates are calculated using a user-specified time interval or block rate. The remaining two quadrants will be discussed later.

Further insight can be obtained by viewing them in three dimensions so that we see the interaction between client, server, and time. Figure 6 is a three dimensional scatter plot of these data. The x-axis is the server, the y-axis is the client, and the z-axis is time. Note there are planes of events in client-server axes. These indicate wide-spread network failures.

Detecting event bursts is fairly simple. A left-hand side to do this would be:

number of events per second > Threshold

The difficult part is determining a reasonable value for the threshold and the block size. We find that this is facilitated by having capabilities to explore (e.g., through animation) different settings for the block size and threshold.

3.4 Periodic Patterns

Our experience has been that a large fraction of the events are part of periodic patterns, typically in excess of 50% of the total events. For example, periodic events are generated as a side effect of many monitoring activities. Discovering such patterns is, in our opinion, an essential part of effective event management.

Our experience in analyzing events of computer networks is that periodic patterns often lead to actionable insights. There are two reasons for this. First, a periodic pattern indicates something persistent and predictable. Thus, there is value in identifying and characterizing the periodicity. Second, the period itself often provides a signature of the underlying phenomena, thereby facilitating diagnosis. In either case, patterns with a very low support are often of considerable

interest. For example, we found a one-day periodic pattern due to a periodic port-scan. Although this pattern only happens three times in a three-day log, it provides a strong indication of a security intrusion.

Unfortunately, mining such periodic patterns is complicated by several factors:

1. Periodic behavior is not necessarily persistent. For example, in complex networks, periodic monitoring is initiated when an exception occurs (e.g., CPU utilization exceeds a threshold) and stops once the exceptional situation is no longer present.

2. Time information may be imprecise due to lack of clock synchronization, rounding, and network delays.

3. Period lengths are not known in advance. Further, periods may span a wide range, from seconds to days. Thus, it is computationally infeasible to exhaustively consider all possible periods.

4. The number of occurrences of a periodic pattern typically depends on the period, and can vary drastically. For example, a pattern with a period of one day has, at most, seven occurrences in a week, while a one minute period may have as many as 1440 occurrences. Thus, support levels need to be adjusted by the period lengths.

5. Noise may disrupt periodicities. By noise, we mean that events may be missing from a periodic pattern and/or random events may be inserted.

To illustrate the kinds of periodicities present in the data shown in Figure 4, consider the situation in Figure 7. Here, we see a pattern present in the stream of events emanating from two AFS servers. In this pattern, there is a "threshold violated" event followed by a "threshold re-set" event (or possibly two "threshold re-set" events). The timing of the latter is fixed at 5 seconds once the "threshold violation" has occurred. Such a pattern suggests that the "threshold reset" events provide little additional information and so one meta-event should be used that combines the "threshold violation" and "threshold reset". An appropriate correlation rule might be

If type="threshold reset"

AND there was a "threshold violated" event from the same host 5 seconds ago

Then combine the two events

3.5 Finding Similar Patterns

Up to now, we have described rule construction based on discovering a single instance of a pattern. Clearly, it is desirable to locate other instances of the same pattern so that a correlation rule can be synthesized from a more complete understanding of the event data. Thus, this example is more about pattern refinement than describing a new pattern per se.

We begin by describing what we mean by a similar pattern of events. To date, our approach has focused on two-dimensional data that extend the visualizations described above. However, many techniques that address higher dimensions are well known (e.g., principal components analysis). Patterns as used here are defined as sets of points in a scatter plot. Typically, the x-axis is time (which is numeric), and the y-axis is categorical, either event type or the originating host. The similarity of two sets of points is quantified based on a distance metric. This metric is the sum of distances for matching points in the two sets. Thus, there are two problems: (a) how are distances between events computed? and (b) how is the correspondence established between points in the sets?

We begin with the first question. The distance between two events is the sum of the distances of its attributes. For numeric attributes, Euclidean distance is used. categorical variables, a binary metric is employed that is 0 if equality holds and 1 otherwise. This simple approach can be augmented by weighting the contributions of individual dimensional attributes, although our results to date have not required this (which may in part be due to the low dimensionally of the comparisons employed).

Now we consider how points are matched between two sets. This is done by exploiting the distance metric in (a). A "starting" point is identified in the reference pattern. Each point in the event history will be considered as a starting point for a candidate pattern. Once a candidate starting point is chosen, then for each point in the reference pattern, we find a point in the event

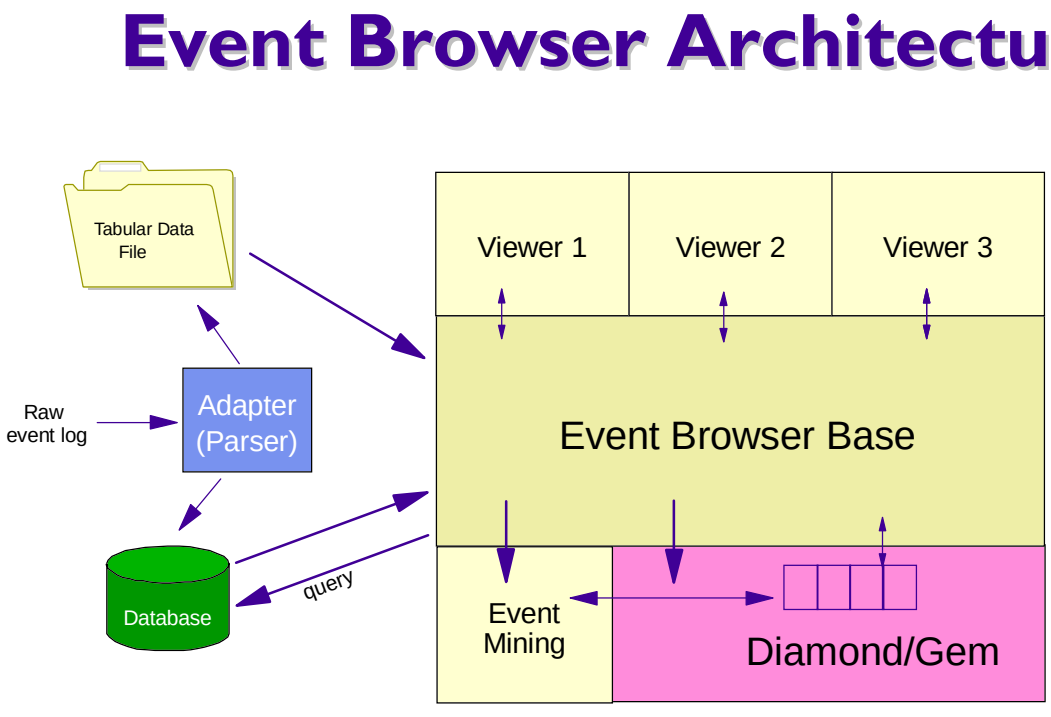

Figure 8: Architecture of EventBrowser

history that has the same distance from the candidate starting point as the reference point does from the reference starting point. We quantify the similarity of the original and new pattern based on the differences in the distances of the matching points from their starting points.

Clearly, the foregoing is computationally intensive. Thus, we exploit several characteristics of to gain efficiencies. These are:

1. The distance metric is cumulative. That is, as more points are matched, their combined pattern distance does not decrease.
2. The search space has low dimensionality. That is, we only consider two search dimensions (e.g., time and event type).
3. The search space is relatively sparse. That is, not all possible pairwise combinations of points need to be considered.
4. The match target pattern is relatively small. That is, only a few points are chosen as for the target pattern.

The actual pattern search procedure involves several steps. First, the searchable field can be narrowed a priori if one of the search dimensions is categorical, as it almost always is in our situations. That is, since the distance metric for categorical data is either equal or not equal, then all data points with categorical *y* not equal to some *y* value in the pattern can be immediately eliminated from consideration. Typically, this reduces the size of the problem by *at least* one order of magnitude.

The search is performed one target pattern point at a time, by simultaneously *searching* through and *pruning* the search space. At each iteration, each point in the searchable space is considered to be part of a potential matched pattern, and its lowest possible and highest possible pattern distance metrics are calculated. Potential patterns can be entirely eliminated if their best possible distance metrics are worse than the worst possible for the best found so far. The acceptable lower and upper bounds for the possible closeness of found matched patterns typically grow toward each other at each iteration. The advantage of such a technique is that the size of the searchable space will diminish very rapidly, and the search will rapidly become more and more efficient (after the slowest first step). The disadvantage is that initial bounds must be specified for the widest acceptable distance metric.

To see how similarity query works, consider the spike patterns in Figure 4. We would ideally like the user to be able to select one (via painting) and then ask the system to find other similar patterns, for example, the second and/or third spikes. This will be discussed in detail in the final part of section 4.

4. Tool for Discovering Event Patterns

The examples discussed in Section 3 motivate the development of a tool to visualize and analyze event data to extract patterns to facilitate the construction of correlation rules. This section describes EventBrowser, a tool for discovering actionable patterns in event data. The discussion addresses the overall architecture of EventBrowser and the design of its graphical user interface.

We begin with the overall architecture, which is displayed in Figure 8. Multiple viewers provide a means of examining event data (e.g., the viewers in Figures 3-6). These viewers interact with the EventBrowser Base, which provides overall control and a componetized user interface. The user interface is based on the very rich components provided by the Diamond/GEM visualization engine [Ra94], [Rat93]. EventBrowser seeks to provide visualizations and pattern discovery for large volumes of event data. As such, data may be stored in flat files and/or database tables, and so appropriate access to database elements are provided. Last, algorithms for mining event data are central to EventBrowser. Exploiting these algorithms requires access to data (as well as using advanced DB facilities that support mining) and appropriate visualizations to study the patterns discovered.

A central consideration in EventBrowser is the synergy between algorithmic approaches to pattern discovery and visualizations. This is evidenced in several ways. The patterns that appear in Figure 4 in the scatter plot are in large part a consequence of choosing a good ordering for categorical data (as in [MaHe99b]). Another more far reaching effect relates to having a uniform GUI for displaying the results of mining algorithms, such as for event bursts or periodic patterns. These latter considerations are substantial and so are discussed in detail.

We begin by discussing the key concepts in the EventBrowser GUI. These concepts emanate from the assumption that we are examining tabular data, that is, data with the same attributes. Greatly aiding us in the development of visualizations of event data is the Diamond [Ra94] visual data analysis package that provides a programmable environment for users to explore and manipulate tabular spreadsheet-like data. Diamond's basic mechanism is to depict data with pictures and to help the user manipulate the pictures to rapidly gain insight into the data. The purpose of multidimensional data exploration and analysis is usually to find and expose relationships among data variables. Diamond allows numerous simultaneous data presentations to be easily compared and contrasted. EventBrowser employs several Diamond visualizations for its event "views". Other standard controls (e.g. buttons, checkboxes, radio buttons) are wrapped with the Diamond pictures to make up an Event Browser **viewer**.

There are many kinds of complementary pictures available which can be freely opened and closed at any time. The various kinds of pictures visualize characteristics of the loaded event data and the relationships between subsets of data. The objects shown in all the pictures are *dynamically linked* through common colors. That is, color painting operations applied through *any* picture in any window will be immediately reflected in *all* of the relevant visible pictures and windows.

EventBrowser's GUI is structured into a number of basic viewers from which complex viewers are constructed. The most commonly used basic viewers are line plot, scatter plots and category tables. While the first two are widely used, the last is new to EventBrowser and so deserve some explanation.

Figure 3 contains an example of two category tables that are displayed side by side. A category table consists of rows of unique event N-tuples and their repetition counts. For example, the left category table in Figure 3 summarizes Host information and the right table lists Event information. The Count field indicates how many events originated from a given host (left table) and how many occurrences of an event were generated (right table). The columns may be sorted by clicking on the headings. The spectrum bars to the left of the count field are painted by EventBrowser to reflect the proportion of hosts that have been painted in another view. So, for example, if an event, e.g. "Arm threshold", was "painted" red by the user, its associated hosts (e.g. hosts that generated that event) would be automatically painted red by EventBrowser and the spectrum bars would give a visual depiction of the proportion of events that were of the specific type ("arm threshold") that were generated by each host.

Many of the complex viewers in EventBrowser are combinations of other viewers. For example, the summary of attributes displayed in Figure 4 consists of two linked category tables. In EventBrowser, this is called an Attribute Viewer. In Figure 5, the temporal data are linked with categorical data by combining a category table and a scatter plot, called a Combination Viewer. When the user paints the cloud pattern green on the scatter plot, the associated events are painted

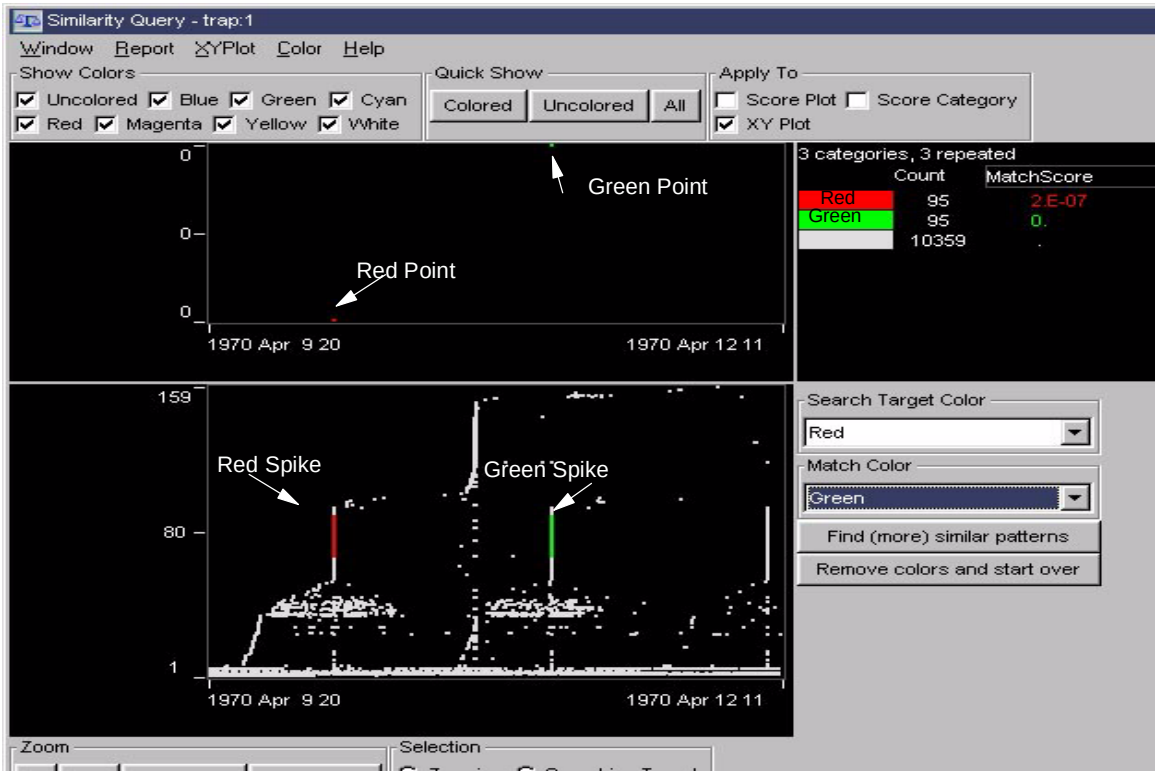

Figure 9: Similarity Query

green in the category table. EventBrowser also provides a combination of two scatter plots, called a Plot Viewer. Event Rate Analysis, shown in Figure 5, uses line plot, scatter plot, and category

table components. By using different colors and painting and filtering colors in views, it becomes easy to explore and uncover patterns in the underlying data.

EventBrowser's most complex windows are its mining viewers: Event Rate Analysis, p-Pattern Detection, and Similarity Query. These viewers attempt to automatically detect patterns in the underlying data. EventBrowser then exposes the patterns visually using windows made up of the single component views described above. A central consideration in EventBrowser is how to specify parameters of mining algorithms (e.g., block size for event rate) and how to explore the results of mining. One GUI construct of importance here is the ability to list patterns and link patterns to category tables, scatter plots, and the other component views. An example of this is shown in Figure 5, Event Rate Analysis, where the bottom right category table indicates the "bursts" and the line plot at the upper left shows the blocked rate for the user-specified block size (x-axis=time and y-axis=blocked rate). The block size is a time interval into which the entire time span is divided and the blocked rate is the number of events occurring in each block. So, for example, if the entire time range is one day, the block size might be set at 15 minutes and the block rate line plot would show, for each 15 minute interval throughout the day, the number of events that occurred. EventBrowser allows the user to specify the block size; it also assists him in selecting a block size by providing block-size animation that can be paused when an appropriate one for the analysis is found. The animation increment is also user-editable. The bottom left quadrant of Figure 5 shows the scatter plot of time (x-axis) and server (y-axis). Note that the burst is not apparent from the scatter plot alone.

EventBrowser's P-Pattern Viewer employs a complex underlying mining algorithm but uses only a single data table view component (tabular, spreadsheet-like view) to display its results. Each row in the data table represents a discovered pattern. The first column gives the pattern interval in seconds. The second column gives the count or number of cycles for the given pattern. Next, the host and event ids are presented in columns three and four respectively. Finally, column five shows the score assigned to this pattern or, the probability that it is indeed a p-pattern. The user may color rows in the p-pattern window and the specific hosts/events that make up that p-pattern will be likewise colored in other existing EventBrowser views. So, for example, if a row in a p-pattern table with an interval of 300 seconds (i.e. 5 minutes) were colored red, an xy plot of events over time would show the p-pattern in red also.

The Similarly Query viewer is a hybrid with both manual and automatic components which allows a user to highlight a pattern of interest and then automatically finds other similar patterns. It uses two scatter plots and a category table, as shown in Figure 9. The bottom left picture is the main scatter plot where the user has selected the target pattern of points to search for by painting them in the search color. After the search has been performed, this scatter plot also identifies the found matched patterns in the match color. The top left picture is a Score Plot of found match scores after a search has been performed. The x-axis is the same as the x-axis in the main scatter plot blow. The colors of the displayed points also correspond. The y-axis on the Score Plot is a match score as determined by the pattern difference metric, where a score of zero or near zero means a perfect or near perfect match, and higher scores mean more dissimilar matches. The top right picture is a Score category table of found match scores after a search has been performed.

To perform and similarity query, the user first selects the target pattern of points to search for by painting them with the *search color* (e.g. red) in the main scatter plot. To work effectively and efficiently, the target pattern painted should be made up of a relatively small number of points. After the points in the target pattern have been colored, the *Find (more) similar patterns* button is pressed to begin the search. The results of the search appear in all three pictures in the *match color* (e.g. green). The original target pattern also appears in all three pictures in the *search color*. The top right Score Category table summarizes the results of the pattern search.

The match scores and point counts for similar patterns (if any) will be in green, and will be listed in order of increasing score (i.e., distance), which is by decreasing degree of similarity. Very low scores indicate relatively good matches, which typically occur when the target and found patterns each contain the same number of points in nearly the same arrangement. Very high scores indicate relatively poor matches, and may occur when the target and found patterns contain different numbers of points, where some subset or subsets of them may have similar arrangements. Sometimes, less similar pattern matches will be listed in gray under the green ones.

The top left Score Plot details the results of the pattern search. It shows the found patterns and their match scores in color and in Time (which is usually the x-axis). The very best matched points with the lowest scores should be at the bottom in red, representing the original target pattern. This graphical representation of the match scores usually provides a good visual indication of how close the found pattern scores really are.

The bottom left scatter plot shows the original data and the results of the pattern search in-situ. The pattern searcher stops looking when there are no more scores close to the range of scores it has already found. However, there may be additional, more dissimilar patterns with scores in a higher range. To find still more patterns similar to the original target, the user can press the *Find (more) similar patterns* button again. If any more can be found, then the pictures will change appropriately.

At any time, the user can repaint the scatter plot to add to or subtract from the original pattern, and then press the *Find (more) similar patterns* button again. Points may be subtracted from the original pattern by temporarily changing the *Search Target Color* (perhaps to *Uncolored*)**,** and repainting one or more of the previously painted points.

5. Conclusion

Event management is a fundamental part of systems management. Over the last fifteen years, automated operations have increased operator productivity by using correlation rules to interpret event patterns. While productivity has improved, a substantial bottleneck remains—determining what correlation rules to write.

This paper describes how visualization and data mining can be used to identify patterns of patterns of interest and to filter events that do not aid in action taking or root-cause analysis. We describe a methodology for systematically discovering correlation rules from event histories. This

methodology iteratively finds patterns, determines actions to associate with the pattern (including filtering its events), and then constructs correlation rules from this information. We present examples of scenarios that employ this methodology, including visualizations and mining techniques that aid in pattern discovery. Finally, we described the architecture of the EventBrowser, a tool that incorporates visualization and mining techniques that aid in rule discovery.

Our future work will address new patterns, scalability considerations, and associated visualizations to aid in both of these efforts. We are currently engaged in customer trials involving ten thousand to one million events per day. The experience should aid us in assessing our methodology and scaling our tools.

References

[AgImSw93] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. *Proc of Very Large Data Bases*, pp. 207-216, 1993.

[AgSr94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *Proc. of Very Large Data Bases*, 1994.

[AgSr95] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proc. 1995 Int. Conf. Data Engineering*, pp. 3-14, 1995.

[ApHo94] C. Apte and S.J. Hong. Predicting Equity Returns from Securities Data with Minimal Rule Generation. *Knowledge Discovery and Data Mining*, 1994.

[ApWeGr93] C. Apte, S. Weiss, G. Grout. Predicting Defects in Disk Drive Manufacturing: A Case Study in High-Dimensional Classification. *Proc. of IEEE Conference on Artificial Intelligence and its Applications*, 1993.

[CoSrMo97] R. Cooley, J. Srivastava, and B. Mobasher. Web mining: Information and pattern discovery on the world wide web. 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997.

[KlMaTo99] M. Klemettinen, H. Mannila, and H. Toivonen. Rule Discovery in Telecommunication Alarm Data, **Journal of Network and Systems Management**, Vol. 7, No. 4, 1999.

[MaHe99a] Sheng Ma and Joseph Hellerstein. EventBrowser: A Flexible Tool for Scalable Analysis of Event Data. *Distributed Operations and Management*, 1999.

[MaHe99b] Sheng Ma and Joseph L. Hellerstein. Ordering Categorical Data to Improve Visualization. *IEEE Symposium on Information Visualization*, 1999.

[MaToVe97] H. Mannila, H. Toivonen, and A ~Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3), 1997.

[Mill86] K.R. Milliken, A.V. Cruise, R.L. Ennis, A.J. Finkel, J.L. Hellerstein, D.J. Loeb, D.A. Klein, M.J. Masullo, H.M. Van Woerkom, N.B. Waite. YES/MVS and the Automation of Operations for Large Computer Complexes. IBM Systems Journal. Vol 25, No. 2, 1986.

[BaNi93] M. Basseville and I. V. Nikiforov, Detection of Abrupt Changes: Theory and Application. Prentice Hall, 1993.

[TaHaHeMa2000] David Tayor, Nagui Halim, Joseph Hellerstein, and Sheng Ma. Scalable Visualization of Event Data. *Fourth IEEE Workshop on System Management*, June, 2000.

[Ra93] David A. Rabenhorst and BMPD Statistical Software, Inc. "BMDP/DIAMOND User's Guide", 1993.

[Ra94] David A. Rabenhorst. Interactive exploration of multi-dimensional data. *Proc. of the SPIE Symposium on Electronic Imaging*, 2179, pp. 277-286, February, 1994.

[YeSkEyYeOh96] S.A. Yemini, S. Sliger, M. Eyal, Y. Yemini, and D. Ohsie. High Speed and Robust Event Correlation. *IEEE Communications Magazine*, Vol 34, No. 5. pp. 82-90, 1996.