GBF (Grouping By Frequency) Correlator

Fukiko Hidano, Vijay Machiraju
Software Technology Laboratory
HP Laboratories Palo Alto
HPL-2001-197
August 9th , 2001*

With large distributed systems, management of generated events become quickly unmanageable. The idea of Grouping By Frequency (GBF) Correlator is to make sense out of millions of events quickly by sorting them into group instances by its number of occurrences. Unlike existing model-based or rule-based systems, minimal information on the network is required. Once the events are grouped, it is analyzed separately to determine the root cause. This paper presents general setup for such distributed system for event analysis and provide detailed description of the GBF Algorithm.

* Internal Accession Date Only Approved for External Publication

GBF (Grouping By Frequency) Correlator

Fukiko Hidano and Vijay Machiraju
Hewlett-Packard Labs
Palo Alto, California

Abstract

With large distributed systems, management of generated events become quickly unmanageable. The idea of Grouping By Frequency (GBF) Correlator is to make sense out of millions of events quickly by sorting them into group instances by its number of occurrences. Unlike existing model-based or rule-based systems, minimal information on the network is required. Once the events are grouped, it is analyzed separately to determine the root cause. This paper presents general setup for such distributed system for event analysis and provide detailed description of the GBF Algorithm.

1 Introduction

With the advances and demands to support various transactions over the Internet, we are flooded with the collection of data that represent various transactions and corresponding events being generated in such infrastructures. For a large network, several million alarm messages (events) are generated on a typical day [GARD 98]. These messages/events do not carry ID's of particular transactions, and therefore it is left for us to decide which data/events are correlated and if possible, to link them to their root causes (events) [HART 95]. Often, the collection of generated events represents the symptoms rather than the cause [GRUS 99], where one main event triggeres various other events that are most likely to contain useless information. Therefore, the root cause is obscured by the symptom events, which complicates the analysis of the collected event data. It is then important to incorporate or design a mechanism (usually referred to as event correlation) to help identify the root cause or the main event.

Event correlation has become an important topic where various techniques have been proposed to isolate the main event of interest. However, the proposed methods often require in-depth human intervention and do not scale/adapt well to changing requirements. In order to alleviate tedious configuration and manual event correlation, we propose a ***Grouping By Frequency*** (GBF) Correlator. The GBF Correlator minimizes the human

intervention in which the analysis of events becomes more manageable and realistic. The algorithm works such that large numbers of events are roughly grouped to create an initial starting point for analysis. With iteration and a small amount of intervention by the system administrator, the system will reach a stable operating point. This report first reviews current approaches then in section 3, the GBF correlator is discussed in detail. Some preliminary simulation results are shown in section 4 and the conclusion is presented in section 5.

2 Past/Current Works

Past work on event correlation can be categorized into several methodologies: *rule-based, model-based* (deterministic v.s. stochastic), *AI-based,* and *case-based* [GARD 96, LEWI 93]. The most commonly used methodology is the *rule-based* approach (e.g. "NetFACT" from IBM [HOUK 95]) in which a set of rules are expressed using a rule-description language. These rules are used by an engine to automatically process the collected events. In *model-based* solutions [KATZ 95], each object that generates failure events is represented as a set of active terminal objects each with the corresponding probability of its failure. These definitions are encupsulated in a more complete system model called the dependency graph, where the dependency from one event to another with a corresponding probability is also defined. The system requires detailed analysis of various objects that are running in the distributed system as well as a complex computation of event dependencies. In a neural network based solutions [WIET 97], the input and output pair has a fixed number of ports where training data sets need to be provided to program the correlator. When the number of input/output pairs changes, not only must the system be completely retrained, but a new internal programming architecture based on the new data sets also needs to be generated.

The shortcomings of the *rule-based* solutions' are that it requires detailed understanding of the network and needs a number of iterations to make the system function correctly. It requires high maintenance as well. Whenever the target system has been re-configured, or new event types have been created, the system of rules needs to be re-examined and re-configured, which is non trivial. The *model-based* approach requires extensive understanding of the system topology and functionality to set up the correlation system. It has the same weakness as the rule-based approach in that setup and re-configuration upon changes in the system are non-trivial. *AI-based* approaches, such as the use of neural networks have been proposed [WIET 97]. However, AI-based approaches have the draw-

back of requiring a long training period before they can deliver any value where if the input and output of the neural net changes, the system must be re-trained from scratch. Traditional *rule-based* approaches can be combined with AI modules such as a neural network pattern discovery system [GARD 98]. The *Case-based* approach tries to overcome this limitation by making the system more adaptable to changes with less maintenance. It achieves this by keeping track of various cases that occured in the past and matching them with the current case. The problem is resolved by applying an appropriate solution from the case library as opposed to creating specific rules or a specific AI module. Adaptation takes place when a new case is resolved and added into the case library (or knowledge base). As in the case of an AI solution, this approach requires a fairly lengthy learning process before it becomes usable. Most systems use a knowledge base to enhance the correlation process(see Figure 1).

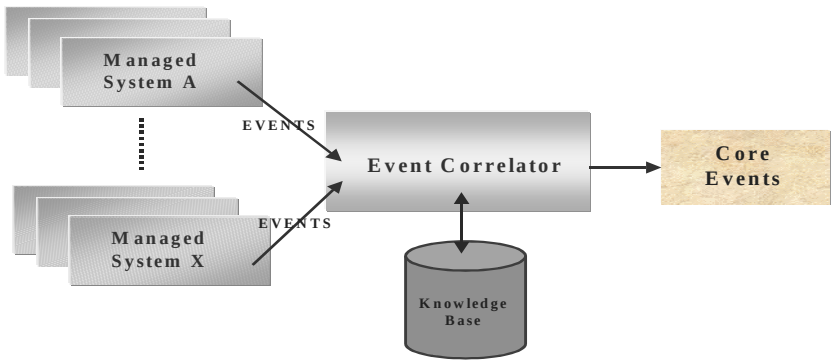

FIGURE 1. General Event Correlator Setup

The proposed GBF solution is flexible and scalable since it should adapt quickly to the addition of new event types or configuration changes, and provide accurate solutions without a requirement for excessive human intervention to re-configure/re-train the system. This is discussed in detail in the next section.

3 GBF Correlator

GBF is a general solution that can be applied to various types of data sets. GBF can best be applied to a scenario in which data (events) to be analyzed have the following characteristics:

- large set is generated
- strong correlations between most events

- recurring

Having a larger set of data helps to identify trends within the data set. Strong correlation between events helps identify groups of related events. The events must be recurring within the set of events collected to help identify the groups.

Given the above conditions, GBF correlation simplifies the correlation process by forming subsets of events that minimize human intervention and are much easier to analyze. Unlike model based systems, the events do not need to be associated with corresponding "objects" that have complex computational requirements for event/dependency probabilities. Unlike case based systems, initial training with the GBF Correlator is minimized, and the internal algorithm is unaffected by changes in the number of input/output pair. The GBF correlator looks at each event as a signal component where the number of event types can grow without impacting the system. Each event class is represented by a signal component with a fixed frequency. Once the events are collected for time period T, they are analyzed and put through a GBF Correlator, which also accesses and customizes the knowledge base. The basic idea is presented in the next section. The GBF algorithm is described in detail in section 3.2.

3.1 Basic Idea

With GBF Correlation, the event correlation process is devided into two layers where the knowledge base can be used to supplement the correlation process. The information in the knowledge base can be modified (add/remove) adaptively to better support the future correlation process. The first part is the GBF correlator which groups relevant events. The second part is the event filter which works on finding the root event that generated the rest of the events (assuming one main event generates various other events that are correlated with the main event). The administrator has an opportunity to examine grouped events and make corrections if necessary to identify the correct root event. The main idea of the GBF correlator is to help reduce the complexity of analyzing a huge set of events or data generated from a large system by grouping them into smaller sets of correlated events. Please see the following figure [Figure 2] that shows the event flow in a generalized distributed system architecture.

Events are collected from all the managed systems and logged into a file. The file is read into the GBF correlator to separate events into event group instances. Each group of

events has a high probability of being “related” in that there is likely one core event responsible for all the events gathered into the group.

Once the events are separated into groups, the events within the groups are analyzed to deduce the core event. If we assume that the events occured “in phase”, meaning that all events collected in a group happened in sequence and that the core event happened first, then the deduction becomes a simple process of selecting the event with the smallest time stamp in the group. If the collection of events are out of phase, meaning that the sequence does not start with the core event, then the degree of phase shift needs to be found by examining the event sequence within the group. This process is done in the Final Event Filter shown in Figure 2. The details of the GBF architecture and algorithm are given in the rest of this section.

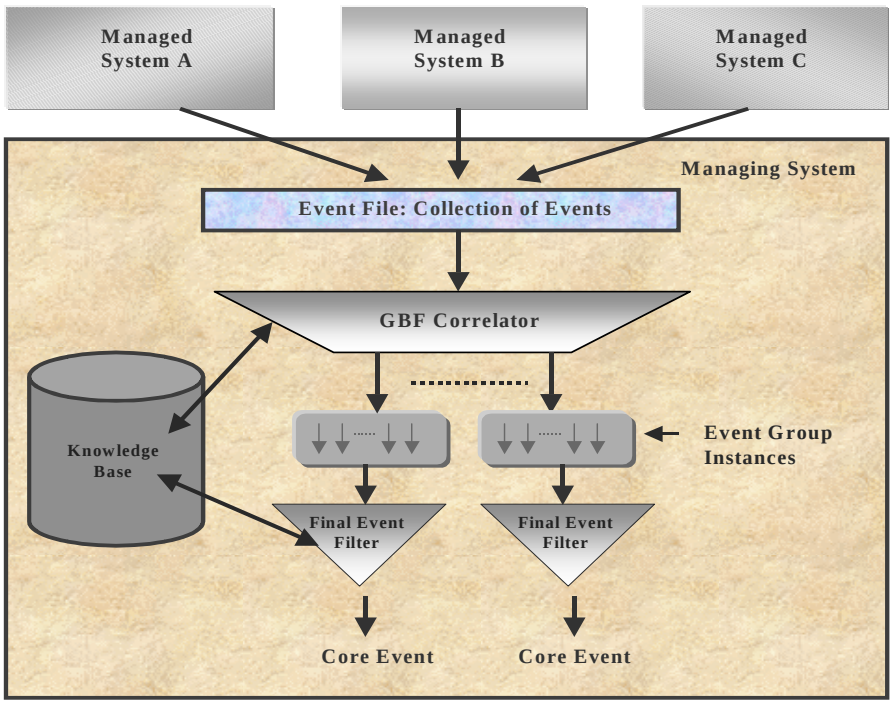

FIGURE 2. Event Flow Diagram

3.2 GBF Correlation Algorithm

The GBF Correlation algorithm consists of two key components. The GBF Correlation System and the Knowledge base. We first describe the main GBF algorithm and then explain how the data contained in the knowledge base can support the algorithm.

3.2.1 GBF Algorithm

We let each event type to be represented by an integer k. Let K be the total number of defined event types in a particular distributed system of interest. Each event type is distinguished by its description and its origination point. For each event type, we also define a function of time t $y_k(t)$ that represents all event occurances of the given type. We collect events for time period T_e, where it is assumed that $y_k(t)$ behaves in a periodic manner with the minimum fundamental period of T_e. We define each event function to be:

$$y_k(t) = e^{jn_k w_0 t} \quad \textbf{(EQ 1)}$$

where n_k is the number of events of type k occurring during period T_e (e.g. the frequency) and $w_0 = (2\pi)/T_e$. Then we let the total event function to be the sum of individual event functions:

$$y(t) = \sum_{k=1}^{K} e^{jn_k w_0 t} \quad \textbf{(EQ 2)}$$

The function $y_k(t) = e^{jn_k w_0 t}$ is an orthonormal function in which each unique frequency n_k forms the new basis function to represent the event function. However, since the value of n_k is not unique to each event type k (e.g. event type 3 can have $n_3 = 23$ and event type 15 can have $n_{15} = 23$) the number of basis functions forming the vector space for y(t) is less than total number of event types K. This will result in some loss of information when we analyze the function in the frequency domain via the fourier transform (e.g. the frequency spectrum will show amplitude but not necessarily indicate which event contributed to it). However, the algorithm keeps track of this information so that it can be used to help aid the grouping process. This is explained in detail later. We now look at y(t) in the frequency domain to help understand why looking at frequency of occurance of these events may help us identify related events. The fourier transform of y(t) is analyzed to help understand the gist of the GBF algorithm. Fourier transform for complex exponential is defined to be:

$$F[e^{jw_0 t}] = 2\pi\delta(w - w_0) \quad \textbf{(EQ 3)}$$

Then the fourier transform of y(t) can be expressed as:

$$F[y(t)] = F\left[\sum_{k=1}^{K} e^{jn_k w_0 t}\right] = \sum_{k=1}^{K} 2\pi\delta(w - n_k w_0) \quad \textbf{(EQ 4)}$$

Figure 3 shows an example of the simple frequency spectrum plot of event function y(t) where N = 8 and n_k for each event type k = 1 to 8 are as follows: n_1=12, n_2 = 82, n_3 = 41, n_4=48, n_5=12, n_6=2300, n_7=8000, n_8=27.

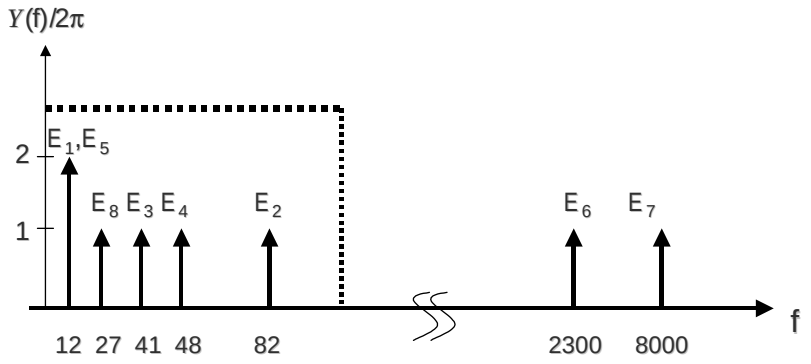

FIGURE 3. Frequency spectrum of event function

As can be seen from the figure, the transform shows frequencies in which events occur. The spectrum doesn't show which events are contributing to the particular frequency amplitude, but if the amplitude is scaled by factor of 1/2π, it will show how many events share that particular frequency. For example, from Figure 3, we see that two event types have the same frequency at 12 events per $T_{e,}$ which happens to be E_1 and E_5.

Some events occur much more frequently than others. This could be because the events are generated on a periodic basis to respond to periodic queries (e.g., license expired for a software), or the events are commonly generated by other events. These events can be noted and reported separately, and are filtered out from the GBF correlation process. The GBF algorithm focuses on low frequency events that are likely to be the main events of interest. The low pass filtering is shown as the dotted line in Figure 3.

Figure 4 shows the grouping process in which events that share the same base frequencies are grouped. In our example, E_1 and E_5 have the same frequency of 12, and therefore are likely to be correlated. It is assumed that either E_1 generated E_5 or E_5 generated E_1. E_4 is

an integer multiple of E_1 and therefore grouped in with E_1 and E_5 to form the group instance 1. Similarly, E_3 and E_2 are grouped as group instance 2.

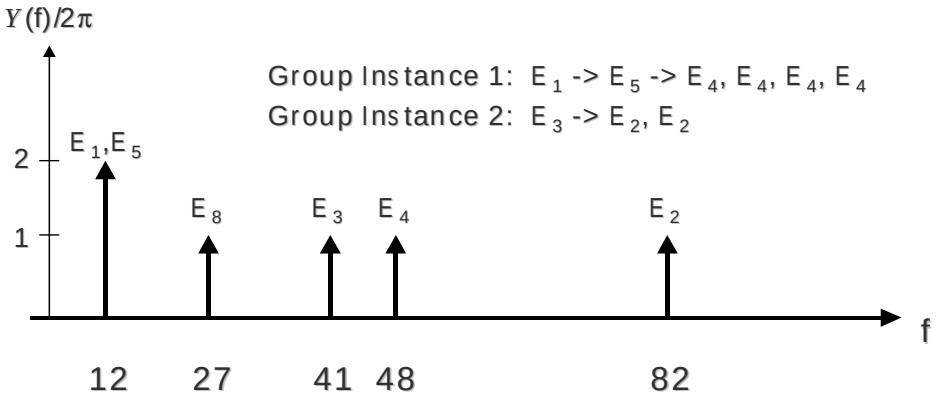

FIGURE 4. The Grouping Process

In summary, the GBF correlator works such that each event is examined and grouped based on its number of occurences during time period T_e. The event set (collection of events generated for time period T_e) is first examined such that the number of occurences within the time period for each event type are counted (e.g. n_k's are identified). After the frequencies of all event types are counted, those with higher frequencies are filtered out from the analysis. This is done because high frequency events are more likely to be symptomatic than causative. Low frequency events are removed from the analysis as well because they are considered to be rare events. Rare events are easier to analyze on their own since the number of occurances is small. However, both low frequency events and the high frequency events are reported to the administrator for review. All the events with corresponding n_k are put into event set. With the example shown in Figure 3, the event set can be represented as $E = \{E_1,E_2,E_3,E_4,E_5,E_6,E_7,E_8\}$.

The algorithm starts working with the lowest frequency event. For example, in Figure 3, E_1 is the lowest frequency event. It goes through the rest of the event set to see if any event contains n_k such that n_k is integer multiple of n_1. Once the events are identified as the integer multiples of the starting event currently under analysis (e.g. E_1), the events are grouped as discussed earlier. This process is repeated until it reaches the highest frequency events unless the pattern is recognized in the event correlation table in the knowl-

edge base (this is discussed in more detail in the next section). The events in the group instance are then removed from the original event set (shown in Figure 5).

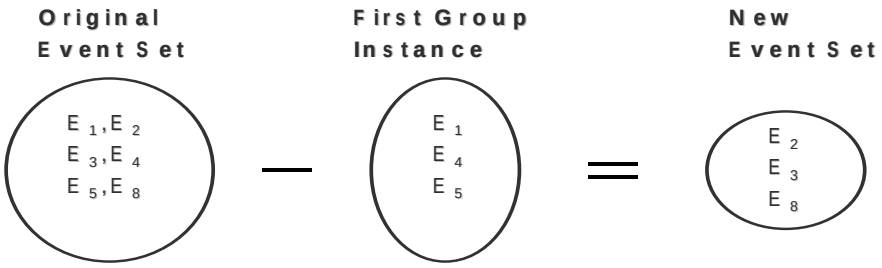

FIGURE 5. Event Set

The grouping process is repeated until there is a “small set” of unidentified events in the event set. In our earlier example, there were no events left after grouping, but in a real situation, there will be events that can not be grouped. At the end of the GBF correlation process, the events in the original event set are grouped into different group instances (see Figure 6).

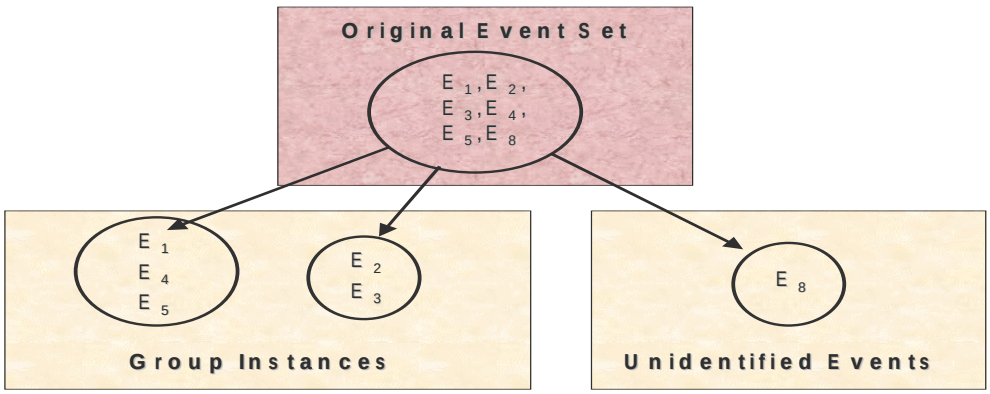

FIGURE 6. Unidentified Events

3.2.2 Out of Phase Event Set

The final event filter then examines the events in each group instance and tries to identify the root cause. The default basic mechanism of the final event filter is to identify the lowest frequency with the smallest time stamp to be the root cause. However, if the events are collected out of phase, then the starting point of the sequence within the group must first be identified. This is illustrated in Figure 7.

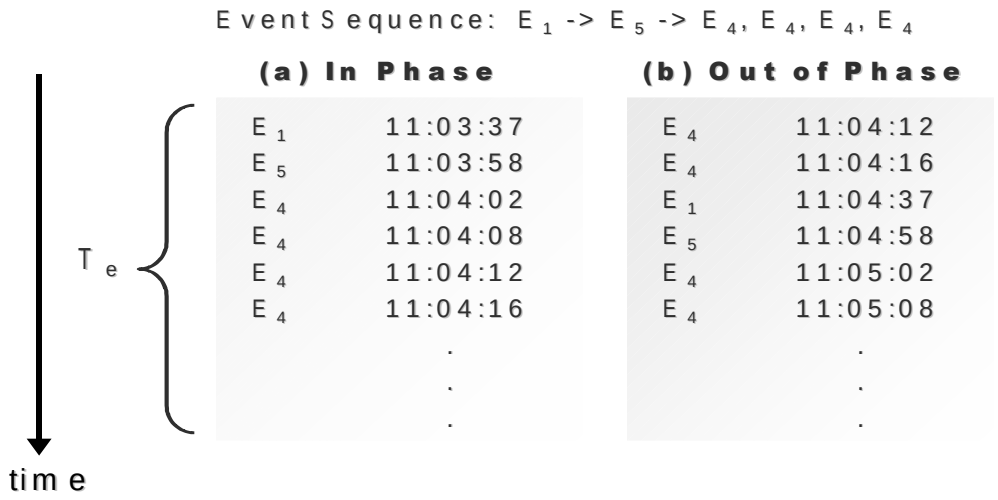

FIGURE 7. Event Files (a) In Phase (b) Out of Phase

In Figure 7, event sequence works such that event E_1 triggers a series of other events (E_1 -> E_5 -> E_4, E_4, E_4, E_4). If the file starts with some event other than E_1, then the seqence is out of phase. The algorithm first needs to find the start of the sequence. The starting event can be found by referencing the event correlation table in the knowledge base as discussed in the next section. The lowest frequency events within the group instance become the candidate for the starting event where timing correlation can be used to accurately identify the starting event.

Another problem with an out of phase event file is that the complete sequence of events may not be captured. For example, the event file can miss the first initiating event and stop before logging the complete sequence.

Therefore, when examining the correlation, it can not be assumed that the frequency of the related events occurs at an exact integer multiple. The offset can be calculated by calculating how many events of the event under examination can occur during one “period” of the root event currently under analysis.

Let event type r is the root event currently under analysis and the event type t is the event under the multiplicity test. If we use the complex exponential to represent the event functions, $T_t = T_e/n_t$ and $T_r = T_e/n_r$ then,

# of events of type t occuring during T_r **(EQ 5)**

But $T_r / T_e = 1 / n_r$, and therefore

$$\text{\# of events of type t occuring during } T_r = \boldsymbol{n_t / n_r} \quad \textbf{(EQ 6)}$$

Intuitively, it makes sense that the we allow the testing event to occur one cycle over the exact sequence. So, the ***offset*** ε used in our algorithm is defined to be:

$$\varepsilon = 2 \cdot n_t / n_r \quad \textbf{(EQ 7)}$$

Therefore, during the examining process, the GBF correlator will pick the lowest event (the cause event) and look for events that have frequencies within integer multiples of the cause event plus ε to form the group instances. This means that the examinng process looks at the inequality below for increasing m:

$$\left| n_t - (m \cdot n_r) \right| \leq \varepsilon \quad \textbf{(EQ 8)}$$

substituting (7), we have

$$\left| n_t - (m \cdot n_r) \right| \leq 2 \cdot n_t / n_r \quad \textbf{(EQ 9)}$$

Let

$$R = n_t / n_r \quad \textbf{(EQ 10)}$$

$$\left| n_t - (m \cdot (n_t / R)) \right| \leq 2 \cdot R \quad \textbf{(EQ 11)}$$

After rearranging, we get

$$n_t \leq (2R^2) / \left| R - m \right| \quad \textbf{(EQ 12)}$$

Substituting (10) and solving for n_r, we have

$$n_r \leq (2R) / \left| R - m \right| \quad \textbf{(EQ 13)}$$

The figure below shows the plot of (13) with n_r as a function of R and m where the inequalitiy is replaced with an equality.

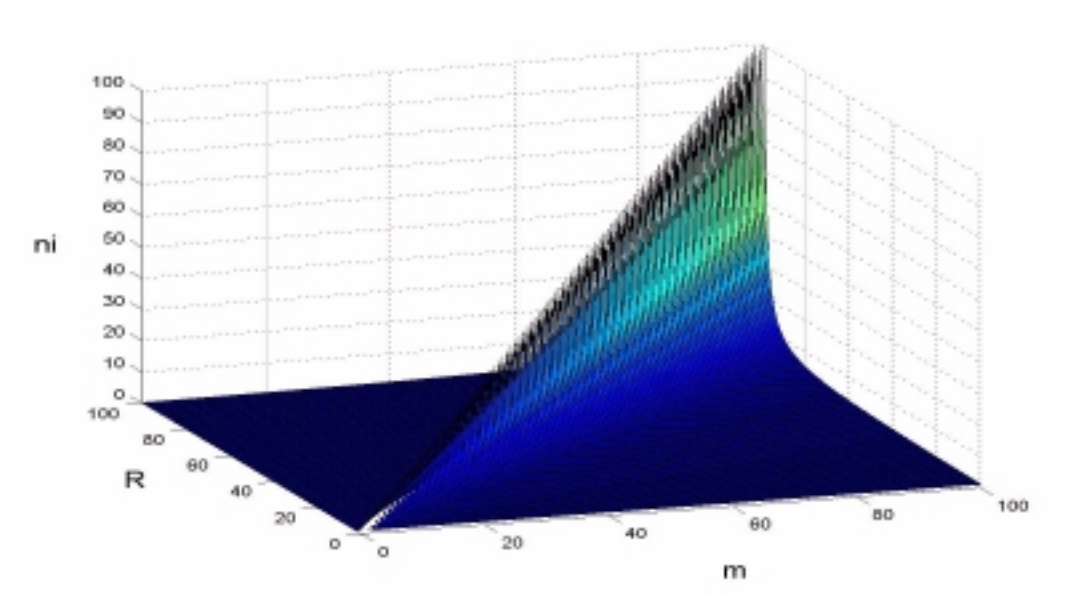

FIGURE 8. nr as a function of R and m

As the figure shows, the function diverges as the ratio R increases. When m = R, n_t meets the requirement of R resulting in left expression to be infinity, permitting the algorithm to include n_t as the group instance. As evident from the figure as well as (12), when m = R-1 or R+1, the integer multiple is offset by 1. In this case, we see that the following set of n_t will be included in the group instance:

$$n_t \leq 2 \cdot R^2 \quad \textbf{(EQ 14)}$$

After substituting (10), we have

$$n_t \geq {n_r}^2 / 2 \quad \textbf{(EQ 15)}$$

(15) means that any n_t that satisfies this condition is included as group instance. In order to make sure that for m = R-1 or m=R+1, above n_t doesn't get included in the goup instance, we need to make sure that

$$n_t < {n_r}^2 / 2 \quad \textbf{(EQ 16)}$$

As we increase T_e, n_r increases accordingly and therefore easy to satisfy condition (16). Hence the algorithm is more accurate as we have larger data set with longer T_e. After substituting (10), (16) can be expressed as

$$R < n_r / 2 \quad \textbf{(EQ 17)}$$

This shows that the algorithm is accurate as long as ratio R is small compared to the number of occurence of event r within T_e. Again, n_r can be made large by increasing T_e.

We can do similar analysis for m = R-2 or m=R+2. The end results that are comparible to (16) and (17) are:

$$n_t < n_r^2 \quad \textbf{(EQ 18)}$$

and

$$R < n_r \quad \textbf{(EQ 19)}$$

Above shows the requirement in which n_t is not included in the group instance for m=R-2 or m=R+2. As you can see the requirement is bit relaxed compared to the requirement for the condition to not include n_t for m=R-1 or m=R+1. The requirements are cumilative in that requirement for R-2 is included in the requirement of R-1 and the requirement for R-3 is included in the requirement of R-2 etc. Therefore, the requirement to not include event which has the integer multiples of m=R-1 covers all of other integer multiples that are not R.

Once the group instances and the probable cause event are identified (by GBF Correlator and Final Event Filter), the results are presented to the system administrator for furthur analysis. The administrator has an option to examine the entire events in the group instance for verification purposes and make appropriate changes to the presented data (e.g. change the cause event to something else that makes more sense or redefine group instances). The corrected information is then registered with the event correlation table in the knowledge base for future references. The use of event correlation table in the knowledge base is discuessed in the next section.

3.2.3 Event Classifications

In this section various event types and the corresponding effectiveness of the GBF Correlator are discussed. Simulation results for each event type are discussed in section 4. We define three different event class:

- Completely Independent Event Sequences
- Shared Event Sequences
- Uncorrelated Independent Events

We first define the ***event group*** for a particular event set E. The event group consists of a collection of events that make the group instances (i.e. Event Sequences) for a particular event set E. We denote each group event to be Sm $m \in 1, M$ where M is the total number of group instances found in event set E. For example, if the first group instance is $\{E_1, E_5, E_4, E_4, E_4, E_4\}$, then the corresponding event group $S_1 = \{E_1, E_5, E_4\}$. The complete event group $S = \{S_1, S_2, ... S_M\}$ and the complete event set in terms of event group is:

$$E = \bigcup_{i \in M} S_i$$

If we let event group S_m to be denoted by event index i:

$$I_a = \{ \quad i \in M \quad : \quad E_i \in S_a \quad \} \qquad a \in [1, M]$$

and assuming that event set E contains events occuring “in Phase”, then the terminology can be used to define the above event classes.

Completely Independent Event Sequences

Completely Independent Event Sequences meet the following conditions:

1) $n_i = kn_j \quad k \in I \qquad \text{for} \qquad i \in I_a \qquad j \in I_b \qquad a = b \qquad a, b \in S$
2) $n_i \neq kn_j \quad k \in I \qquad \text{for} \qquad i \in I_a \qquad j \in I_b \qquad a \neq b \qquad a, b \in S$
3) $I_a \cap I_b = \{\varnothing\} \qquad \text{for} \qquad a \neq b \qquad a, b \in S$

For example, if $S=\{S_1, S_2\}$ (e.g. only two event group) and if we have the following event sequences:

Sequence 1: E_1 -> E_2 -> E_3

Sequence 2: E_4 -> E_5

then we see that $I_{s1} = \{1, 2, 3\}$ and $I_{s2} = \{4, 5\}$ are mutually exclusive meeting the third condition: $I_{s1} \cap I_{s2} = \{\varnothing\}$

Shared Event Sequences

Shared Event Sequences inherit the condition 1 and 2 given for the completely independent sequences for all the non shared events. The set of indexes for the Shared events can be defined as:

$$I_{sh} = \bigcup_{a \neq b} (I_a \cap I_b) \qquad a, b \in S$$

Where $I_{sh} \neq \{\varnothing\}$. For example, if $S=\{S_1, S_2\}$ (e.g. only two event groups) and if we have the following event sequences:

Sequence 1: E_1 -> E_2 -> E_3

Sequence 2: E_4 -> E_2 -> E_5

then we see that $I_{s1} = \{1, 2, 3\}$ and $I_{s2} = \{4, 2, 5\}$ has the shared event E_2 and therefore, $I_{sh} = I_{s1} \cap I_{s2} = \{2\}$.

Uncorrelated Independent Events

Uncorrelated ramdom events have a set of events such that events do not occur in patterns with respect to other events. Such events are generated independently of other events and are therefore completely uncorrelated.

GBF Correlator works best with completely independent event sequences. For the shared event sequence, the GBF Correlator can currently identify group instances except for the shared events. However, with trend analysis and feedback, this can be worked on. GBF Correlator can not be applied to Uncorrelated Independent Events. This grouping will not make any sense in this case.

3.2.4 Knowledge Base Data Representation

The knowledge base assumed in this report is a very simple set of data. All the data in the knowledge base is kept in a disk file. The table is called an ***event correlation table.*** It contains possible group instances, corresponding core event(s), and the probability of each group instance occuring based on the past occurences. The following figure will clarify

the table structure. The table consists of possible group instances with their core event(s) and their probabilities of occurence.

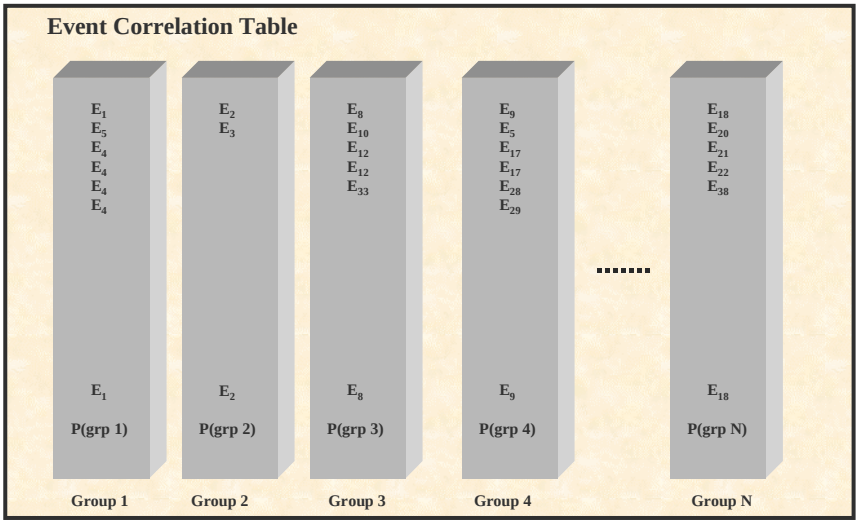

FIGURE 9. Event Correlation Table in the Knowledge Base

The event correlation table is created when the GBF algorithm goes through the grouping process. The group instances in the event correlation table are referenced to help identify group instances in the event set during the grouping process as well. During the grouping process, the initial group members are sent to the knowledge base in order to search for the pattern. When the pattern is recognized, the events composing the event group are subtracted from the event set. The detailed algorithm of the pattern matching is beyond the scope of this report and is therefore not discussed.

4 Simulation

This section discusses some simulation results. The simulation uses an event set file corresponding to the three event classes described in the previous section: independently sequenced events, shared sequenced events, and uncorrelated independent events. The purpose of the simulator is to show how the GBF Correlator can support the grouping of events to help analyze large event sets. The GBF Correlator system is written completely in Java, where sample points have been manually created for each event class. The basic algorithm is shown below:

```
for (int i = 1; i <= K; i++) {
   let i be the first group instance member
   for (int j = i+1; j <= K; j++) {
      for (int k = 1; m < maxMultiple; m++) {
         if ( abs(n_j - m*n_i) <= 2*(n_j/n_i) ) {
            add j into group instance
         }
      }
   }
   proceed to the next group instance
}
```

where events are ordered such that lowest events has the smaller event id (e.g. i for n_i). K is the total event types and maxMultiple is the maximum integer multiple we will be interested in relating various events. maxMultiple has been set to 10 in this simulation.

Completely Independent Event Sequences

For the completely independent Event Sequences, the event file was created with the following event sequences:

Sequence 1: E_1 -> E_2 -> E_3

Sequence 2: E_4 -> E_5

Sequence 3: E_8

The event file was also out of phase such that sequence 1 started with event 2. and as can be seen from the output, E_1, E_2, and E_3 are not exact integer multiples of each other. However, the algorithm accounted for the offset e based on the proposed calculations discussed in section 3, and therefore was able to identify the event sequence in the form of a group instance.

```
event 1 occurred 138 times
event 2 occurred 280 times
event 3 occurred 280 times
event 4 occurred 364 times
event 5 occurred 364 times
event 8 occurred 112 times
event 9 occurred 1 times

Group Instance 1 = E1, E2, E3
```

```
Group Instance 2 = E4, E5
Group Instance 3 = E8
```

Shared Event Sequences

In this event file, the event sequences were made such that E_2 was an event shared by event sequence 1 and event sequence 2 as shown below:

Sequence 1: E_1 -> E_2 -> E_3

Sequence 2: E_4 -> E_2 -> E_5

```
event 1 occurred 275 times
event 2 occurred 1265 times
event 3 occurred 715 times
event 4 occurred 715 times

Group Instance 1 = E1, E3
Group Instance 2 = E2
Group Instance 3 = E4, E5
```

The GBF Correlator can find the non-shared events within the sequence. However, the shared-event E_2 is recognized as an independent Group Instance. It is highly likely that the shared events are not that helpful in determining the cause event and therefore can be eliminated from the analysis anyways.

Uncorrelated Independent Events

In this simulation, events are generated from a pseudo number generator where all events are uniformly distributed.

```
event 1 occurred 244 times
event 2 occurred 255 times
event 3 occurred 236 times
event 4 occurred 224 times
event 5 occurred 248 times
event 8 occurred 231 times
event 9 occurred 238 times

Group Instance 1 = E1
Group Instance 2 = E2
Group Instance 3 = E3, E9
```

```
Group Instance 4 = E4
Group Instance 5 = E5
Group Instance 6 = E8
```

As you see from the results, the frequencies are very close across all the events. This is because the occurrences are uniformly distributed. This is not a realistic scenario, however this can be viewed as the worst case scenario in which the frequencies for all events are similar and therefore really test the robustness of GBF Correlator. As the results show the GBF Correlator recognizes that the events are independent. Though it is not perfect and we see E3 and E9 are both grouped together as group instance 3. However, examining the event file and looking at its sequence, it will become evident that the E3 and E9 are uncorrelated.

5 Summary and Conclusion

In this report, we have proposed an algorithm that will make an analyses of large sets of data more manageable. GBF Correlator takes in a large set of events and groups them into group instances based on the number of occurrence for each event type (event frequencies). When events are correlated, it is highly likely that the correlation trend will be evident in their event frequencies. Reducing events or messages into smaller subsets makes it easier to do analysis and manage events. The final event filter is responsible for examining the group instances and determining the cause event. Correlation in the time domain can be used to make sense of the events. The GBF Correlator works best on large numbers of events where events are highly correlated and recurring. Simulation results show that GBF Correlator can work best on Completely Independent Event Sequences. Currently, GBF Corrlator is unable to map shared-events into each of its group instances. However, with further analysis this can be accomodated as well. GBF Correlator is easy to implement and incorporate into existing systems. Compared with other proposed approaches, it requires minimal learning and minimal maintenance. Though the simulation results show the robustness of GBF Correlator in terms of separating independent events and recognizing the event sequences, with current basic implementation, the results may be inaccurate in realistic systems where event sequences include shared events. However, with the introduction of a knowledge base that gives reliable feedback to the algorithm and some analysis on the timing of the event occurrences, the GBF Correlator can be made to provide an accurate analysis of the event sets.

Acknowledgement

We would like to thank Ed Katz who helped us understand various concepts in the field of AI such as neuronet and fuzzy logic. We would like to thank Klaus Wurster for helping us understand some of the management product for Openview and giving us some sample point for the generated events from a real system. We would like to thank Steve Hoyle for giving us feedback on this document. We would like to thank Aad van Moorsel for guiding us through the convergence analysis of the algorithm. We would like to thank members of our Department for various helpful feedback.

References

[GARD 96] R. Gardner and D. Harle, “Methods and Systems for Alarm Correlation”, Proceedings of Globecom 1996, London, November 1996, P136-140.

[GARD 98] R. Gardner and D. Harle, “Pattern Discovery and Specification Techniques for Alarm Correlation”, Network Operations and Management Symposium, New Orleans, USA, February 1998. IEEE P713- P722.

[GRUS 99] Boris, Gruschke, “A New Approach for Event Correlation based on Dependency Graphs”, Department of Computer Science, University of Munich Technical Report.

[HART 95] I. Rouvellou and G. Hart, “Automatic Alarm correlation for Fault Identification”, Proceedings of Infocom 1995, Boston, April 1995, P553-561 Vol II.

[HOUK 95] Houk K., Calo, S., Finkel, A. (1995). Towards a Practical Alarm Correlation System. In: Sethi, A., Raynaud, Y., Faure-Vincent, F. *Fourth International Symposium on Integrated Network Management*,San Francisco,1995. London, Chapman & Hall, 226-238.

[KATZ 95] Irene Katzela and Mischa Schwartz, “Schemes for Fault Identification in Communication Networks”, IEEE Transactions on Networking, 3(6): 753-764, December 1995.

[LEWI 93] L. Lewis, “A Case-based Reasoning Approrach to the Resolution of Faults in Communications Networks”, Third International Symposium for Integrated Network Management April 1993. P671-P682.

[WIET 97] H Witgrefe, K. Tuchs, K. Jobmann, G. Carls, P. Frohlich, W. Nejdl, S. Steinfeld, “Using Neural Networks for Alarm Correlation in Cellular Phone Networks”, University of Hannover.