Fuzzy Event Correlation Algorithm in Wide Telecommunication Networks

Jacques-H. Bellec and M-Tahar Kechadi
School of Computer Science & Informatics,
University College Dublin, Belfield, Dublin 4, Ireland.
Jacques.Bellec@ucd.ie, Tahar.Kechadi@ucd.ie

Abstract

This paper presents an efficient clustering algorithm for faults identification in large telecommunication networks. The alarms and faults in telecommunication networks present some interesting characteristics like storm and cascade of events. For instance, a single fault may result in a large number of alarms, and it is often very difficult to isolate the true cause of a fault. Our algorithm is especially designed for the event correlation problem taking into account comprehensive information about the system behaviour. Our technique is tested and compared with some available clustering algorithms on some samples from both simulated and real data from Ericsson's network.

1 Introduction

Telecommunication networks are growing in size and complexity at a very rapid rate, and therefore their management is becoming more and more complicated. Each network element can produce a large amount of alarms when a fault occurs. More precisely, when a fault occurs, network devices or components can send messages (alarms) to describe the problem that has been detected. But they only have a local view of the fault, and therefore cannot describe the fault, but just its visible consequences. Moreover, these alarms are very different due to various types of network components involved (such as new equipments, software updates, etc.). The telecommunication network management system is responsible for recording the alarms generated by the network nodes or components and presents them to the operator. However, in large systems, due to the large volume and the fragmented nature of the information contained within these alarms, it is not always possible to locate and solve the faults within a reasonable time. In addition, due to the complex nature of these networks, a single fault may produce a cascade of alarms from the affected network elements and also, a fault can trigger other faults, for instance in the case of overloading. Even though failures in large communication networks are unavoidable, quick detection, identification of causes and resolution can make systems more robust, more reliable, and can ultimately increase the level of confidence in the services that they provide [1].

Alarm correlation is a key issue in a network management system as it is used to determine the faults' origin, and to filter out redundant and spurious events. The alarm correlation systems generally combine causal and temporal correlation models with the network topology. The efficiency and robustness of the models used and the algorithms developed vary from system to system but none of them have yet succeeded to provide a good solution to this problem [2]. In general, data mining techniques are well adapted for analysing collections of data and extracting hidden information. However, the complex nature of the data generated by a wide telecommunication network and the lack of real information contained in an alarm, make unsuitable most of the existing techniques [3].

In [4], we introduced the Behavioural Proximity Technique (BP), which is a hierarchical 3-stage process. The first stage is the pre-processing phase, which includes cleaning and filtering. This is followed by the event recognition process where alarms are gathered according to their behaviour to form events. Finally, these events are correlated to form clusters using an efficient algorithm. As a result, only the important seeds of global events are presented to the network operator, to help identifying and solving the faults in the network.

Here, we propose a new clustering algorithm to deal with the events generated by telecommunication networks and mainly to deal with some issues in the alarm datasets. It is called Fuzzy Event Correlation or FEC_k. The algorithm is characterised by a parameter k; the number of clusters calculated according to the most relevant alarms. In the pre-processing phase, alarms are gathered into events, which are then evaluated according to a value-function $V(e_i)$, where e_i is an event. The idea is to use the events that have higher value-function $V(.)$ as root of the event correlation. It incorporates a fuzzy core, which provides more flexibility to the algorithm.

The paper is organised as follows. In the next section, we describe the fault recognition problem with its constraints and requirements. In section 3, briefly we present our alarm correlation framework, called Behavioural Proximity. Then, we describe our new clustering algorithm, highlighting its key features. In section 4, we present some experimental results and study the performance of our algorithm compared with very popular and widely used algorithm. Our results are obtained on both simulated and real-world datasets. We conclude in section 5.

2 The Fault Recognition Problem

In the past, the network fault management task was performed by human experts. The size and complexity of today's networks, however, have heightened the required level of human intervention so much, that it makes it prohibitive. Therefore many systems have been proposed that integrate event correlation engines to address this issue [5, 6]. The problem of an automatic identification of correlated events has been tackled from various perspectives with different techniques, but none of them has succeeded to achieve a good level of satisfaction for both the network operator and the users [7]. This problem has been proven to be NP-complete and different heuristics methods that have been used have not performed reasonably well.

Many challenges have to be taken into consideration when solving this problem. In the following we present three key issues in modern networks. Firstly, wide networks are rapidly evolving with heterogeneous hardware and configuration policies, making it very hard to get a global view of the network topology of interest. So topological information is partially known. Secondly, the information contained in the alarms is often incomplete or misleading. The alarms are generated by the components in error-state affected by the presence of a fault nearby, but that error-state usually can have several causes or a fault can generate a chain reaction of faults and therefore masking the original cause. The message embedded in an alarm is not always explicit about the fault; it just records some local information about the parent device or component. Moreover, the fault can be categorised as unknown by the components as only few error messages are possible. In this case it does not provide any information about the nature of the fault. Sometimes the network can become overloaded due to some unsolved faults or/and a peak of activity. Some alarms can be destroyed or lost in the network and therefore they are not recorded in the log files. Thirdly, the presence of heterogeneous network components (hardware or software) creates the problem of how the alarms should be interpreted, as different devices have different configuration policies and different ways of issuing alarms for the same abnormal events. For instance, some sub-networks are configured with some particular policies, so they can produce different alarm messages for the same fault. All these issues introduce more difficulties in analysing and mining historical data of these networks. They aggravate the problem of noisy and incomplete datasets. Therefore, a good fault recognition technique should have a very efficient pre-processing and cleaning phase and robust mining/analysis phase. In this case, a more appropriate output of a fault recognition technique would be a set of clusters representing highly correlated events of alarms and giving

some information about some hypothetical faults. A good clustering technique should find clusters that provide enough information about the origin of the faults to the network operator. The main objective is to build some well-defined clusters within a reasonable short-time period and to provide more accurate results if required.

Temporal data mining techniques have different constraints and objectives [8, 9, 10]. They are usually dealing with mining large sequential datasets, which means that the data is ordered with respect to some attribute or dimension. The ordering among the events and data values is so important and crucial in data modelling. For example, fault datasets consist of a recorded series of time stamped events. However, the time is not the only dimension that can be used for indexing the data in alarm network data as the notion of time is not global (which means the time is not exactly the same in each network node). Other dimensions are required to extract a global ordering, which is usually based on some complex relationships between alarms' behaviour and network faults and topology.

Existing Model-based approaches aim to represent the interrelations between the components of the network [11], or the causal relations between the events in the network [12], or a combination of the two [13]. Unfortunately, as the network topology is not known, these models are not suitable for the requirements of the problem at hand. Rule-based [14] and code-based [15] systems also model the relations between the events specifying the correlations according to a rule-set or codebook. Rules can be generated automatically and must be validated by experts. Other techniques, such as neural networks [16] or decision trees, have also been applied to this problem. These approaches vary in the level of expert knowledge required to train the system. Some neural networks, for example, do not require any expert supervision but need to be previously well trained.

In [17] they used association rules and frequent episodes to discover alarm patterns, which were subsequently used in the development of alarm correlation systems. However, their methods do neither capture the notion of alarm similarity nor the uncertainty of the network. Furthermore, association rules and frequent episodes have the drawback of generating many non-interesting and redundant alarm patterns [7].

Alarm clustering approaches aim to discover the hidden patterns in partitioning the data into several non-predefined groups [6, 18]. Many clustering algorithms have been defined to deal with general data. They can be hierarchical, partitional or mixed. Some of them like CURE [19] and BIRCH [20] have a complexity in time and space of $O(n)$. The number of clusters can be static (given as input) or dynamic (calculated automatically). The quality of the clustering varies from an algorithm to another when dealing with non-spherical shapes as shown in [21]. To represent a cluster in the partition process, one or several points can be chosen. The policy about the representation can deeply influence the quality of the results. For some algorithms like k-mean and k-medoids, k scattered points must be initially chosen among the data for the partition. Unfortunately, whereas the initial selection of medoids is a key step of the whole procedure, it is randomly done in most of applications, and often produces poor results. Multiple runs can address this problem in selecting more accurately the best centroids, but it increases dramatically the complexity of the algorithm. Some algorithm can be quite hard to tuned up as they require many input parameters. Their optimisation can only be achieved after many runs which is a major drawback when in presence of a time constraint. Some of the algorithms like CURE and DBSCAN [22] handle outliers well, identifying them from other data points. But the outliers can be interesting as they do not always represent noise. In partial clustering, outliers can be irremediably lost, even if they are interesting for the application. The requirements of our problem is not fulfilled with the available techniques. So we need to imagine a new hybrid technique which takes the best of each of them and combines the skills to achieve good performance. In the next sections, we will present our framework called Behavioural Proximity and explain the effectiveness of our new event-clustering algorithm.

3 The Behavioural Proximity Approach

The Behavioural Proximity Approach is developed for fault management in telecommunication networks. It takes as input any datasets describing the alarms generated by the network components in error-state over time, and it returns a set of critical alarms that should correspond to critical faults. As shown in Figure 1, this technique is composed of four important steps. In the next sub-section, we describe the first two steps as they are part of part of the data pre-processing. The following sub-section is dedicated to the next last two steps, which are the heart of this technique.

3.1 The Pre-processing Phase

The raw datasets contain all the log alarms produced by the network management system. They may contain many duplicates, alarms of wrong format, delayed alarms, etc. The first step consists of eliminating all wrong formats and duplicates. This will only improve the data quality as their is often large. The remaining alarms are then grouped into sets. Each set contains alarms that share the same content. The content of an alarm is the message and the source id.

Some sets contain alarms that occur periodically. So, we introduced the concept of periodic and aperiodic alarms. This is very important for measuring the homogeneity of a set, as well correlated alarms (homogeneous set) can give valuable information about a fault. We consider that each set represents the abnormal behaviour of a device or component during a certain period t. A set of periodic alarms is aggregated into what we call an *event*. For the set containing aperiodic alarms, another treatment is required before grouping them into events. We use the Score-Matching (SM) algorithm to extract periodic alarms from each aperiodic set [4]. This process consists of either inserting some missing alarms to form periodic set or divide the set into sub-sets of periodic alarms.

At this stage all the alarms are grouped into events. Each event has a representative alarm and some other parameters such as the period, attributes inherited from the alarms, etc. So, we consider that each event e_i has a set of m attributes $\{t_1, t_2, \cdots, t_m\}$ and we introduce an evaluation function $V(e_i)$ to assign to an event e_i a value based on its set of attributes. This function can be formalised as follows:

$$V(e_i) = \sum_{j=1}^{m} \alpha_j \omega_j \tag{1}$$

where ω_j is a score (value) of the attributes t_j, usually defined by the network operator and α_j is a scaling factor defining the weight of a given attribute within the whole set of attributes. There are attributes with their relative scores which are common to all the telecommunication networks as shown in [3], but they can be updated to include some particular requirements. The next goal is to determine whether an event is critical or not. So, we use the notion of fuzzy sets to determine this. More precisely, we use a fuzzy membership function $\phi()$ to get the corresponding confidence value of an event. $\phi()$ is based on the S-function [23] where a, b, and c correspond to the minimum score, the average score and the maximum score respectively.

$$\phi(e_i) = \begin{cases} 0 & V(e_i) \leq a \\ 2\left(\frac{V(e_i)-a}{c-a}\right)^2 & V(e_i) > a \text{ and } V(e_i) \leq b \\ 1 - 2\left(\frac{V(e_i)-c}{c-a}\right)^2 & V(e_i) > b \text{ and } V(e_i) \leq c \\ 1 & V(e_i) > c \end{cases} \tag{2}$$

3.2 FECk Algorithm

The goal of this phase is to cluster correlated events. Many clustering algorithms are available in the literature and each of them has some advantages and disadvantages depending on the domain and the constraints applied. We present here some of the requirements for a clustering algorithm

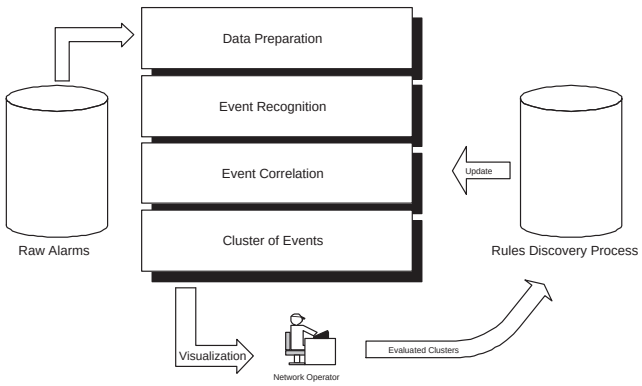

Figure 1. The Behavioural Proximity Technique

that can be used in our system. These requirements are fast response time and high quality of the results. We need a fast algorithm to allow the operator to make quick decision on the location and maintenance of the faulty components. The quality of clustering affects directly the decision of correctly identifying the type of the fault and therefore the network component to be repaired or replaced. So, the operator needs a few high quality clusters. In other words, a cluster has to present enough information by its content to be relevant and useful for the network operator.

The proposed algorithm is called Fuzzy Event Correlation ($FECk$). It takes into consideration both the application and the system requirements; specific behaviour of the alarms, network topology, etc. In the following, we present the core of the algorithm given below (1).

The algorithm has three main steps. The first step, line 4 to 8, identifies the most important events. They are called *primary* events and they are considered to be independent and not related. The primary events are determined using a certain confidence value α, which is fixed by the user. Let A be the number of primary events and let p_i be the i^{th} primary event. Note that A corresponds to the minimum number of clusters that can be returned by the algorithm. The second step constructs the clusters around the primary events. The algorithm calculates the correlation between each non-primary event and every primary event. A non-primary event is assigned to a cluster of a primary event with which it has the highest correlation. The correlation function $\psi_{p_i}()$ is a membership function of an event being in a cluster of the primary event p_i.

$$\psi_{p_i}(e_j, e_k) = S(M_h(e_j, e_k)) \tag{3}$$

where $M_h(e_j, e_k)$ is the Mahalanobis distance.

The second step is the clustering phase from line 14 to 20. Non-primary events are gathered with their closest primary p_i event according to the correlation function $\psi_{p_i}()$. If a non-primary event is not in the neighbourhood of any primary event then it is put in a `bin-cluster`. The correlation function ψ_{p_i} is the membership function of the primary event p_i and defines its neighbourhood. In other word any non-primary event which is located in the neighbourhood of a primary event p_i will be part of the cluster of p_i. Note that an event cannot be in two different clusters. If an event is equally correlated to two primary events, it will be assigned to only one of them.

The third step of the algorithm, line 22 to 27, deals with the weak correlation problem. For instance if an event is correlated to a primary at best at 10%, either it is an outlier or it is a member of a cluster with a different density. In the latter case, the algorithm is executed again with these non-assigned events located in the bin-cluster. We consider only two passes to handle dense and sparse data, but it can be extended to multiple passes if needed, namely if the size of `bin-cluster` is still large. This way, it overcomes the problem of different overlapped densities which is a crucial problem in DBSCAN. After the second round, the remaining events in `bin-cluster` are clustered into smaller `bin-clusters`.

The number of clusters k produced by the algorithm is the sum of all the identified primary

Algorithm 1 FEC_k Algorithm ;

1: INPUT:
2: $E = \{e_1, e_2, ...e_n\}$: Event Set, GDoT α,
$V(e_i)$: Value-function $V(e_i) = s_i$ // scores each event
$\phi(s_i)$: Fuzzy value-function of the primary set ω,
$\psi_{c_k}(e_i, e_j)$: Fuzzy correlation function of two events e_i, e_j in c_k,
3: OUTPUT: C: Cluster Sets
4: **for all** $e_i \in E$ **do**
5: **if** $\phi(V(e_i)) > \alpha$ **then**
6: $e_i \in \omega$, // assignment to the primary set
7: **end if**
8: **end for**
9: $initialize_cluster(\omega)$// cluster initialisation
10: $k \geq |\omega|$, // k clusters will be created at least
11: **for all** $e_i \notin \omega$ **do**
12: $maxCorr = 0$
13: **for all** $e_j \in \omega$ **do**
14: **if** $\psi(e_i, e_j) > maxCorr$ **then**
15: $l = j$
16: $maxCorr = \psi(e_i, e_j)$, //correlation phase
17: **end if**
18: **end for**
19: **if** $\psi(e_i, e_l) \geq \alpha$ **then**
20: $e_i \in C_l$
21: **else**
22: $e_i \in TempSet$ //weak correlation
23: **end if**
24: **end for**

events plus the number of unclassified events. We have $k = \sum_n^p |\omega| + |bin\text{-}clusters|$.

As stated above, we assume that there is no correlation between primary events. This has two important advantages: the first advantage is the complexity. In fact with n events the cost to calculate all distances between each pair of events is of complexity $O(n^2)$ which is too high when considering several hundreds thousands of events. The second advantage is that the relevant representatives, which one of the biggest problems with some clustering algorithms are efficiently determined. This is done once, at the beginning of the algorithm. In addition, if there is a correlation between two primary events, then the two generated clusters will be very close and the user can easily identify them. This will not affect the quality of the results. The detection of correlated primary events will be dealt with in the future.

To illustrate how the FEC_k algorith works we consider the example given in Figure 3.2. The first step is to identify the main events by using a score function $V()$ followed by a fuzzy function $\phi()$. The events with scores that are above a minimum confidence value are chosen to be primary events (presented in solid circles in the Figure). Secondly, the correlations are calculated between any pair (primary - non-primary) events. Any non-primary event which is located within the neighbourhood of a primary event will be part of that cluster; (as shown in the Figure). Note that some non-primary events are not located in the neighbourhood of any primary event. These events are considered to be noise or outliers and, therefore, they are are gathered in `bin-cluster`. The third step of the algorithm will partition the `bin-cluster` into smaller sub-`bin-clusters` if needed.

We can notice that FEC_k shares some common features with PAM [24] and DBSCAN. Moreover,

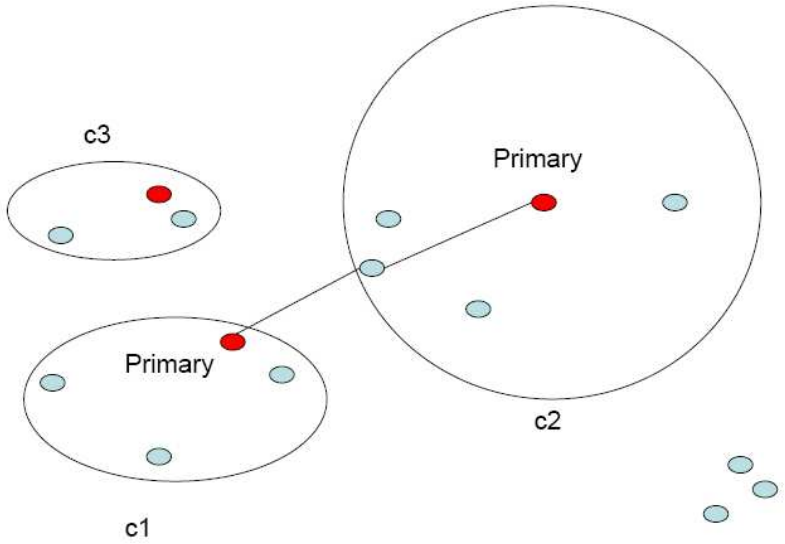

Figure 2. Example with a small set of events

the pre-processing phase can directly affect the quality of the results generated by our algorithm. The main features of the algorithm are as follows:

1. It uses fuzzy set theory to group the events and it produces a complete partition of the data. The number of groups or clusters (k) is generated automatically. It outputs critical clusters.
2. It handles outliers by using a minimal correlation degree calculated from the distribution of all available correlations.
3. It overcomes the sparse and high-density data problem. It has a complexity of $O(kn)$ with k very small compared to n, which is the number of events.

In the next section, we will present an evaluation of our algorithm using simulated and real data sets.

4 Performance Evaluation

This section presents the performance evaluation of FEC_k with simulated and real-world data collected from a wide-area telecommunication network. To use two traditional metrics to measure the accuracy of the clusters obtained by this technique. These measures are as follows:

$$Recall = \frac{\text{Relevant Events clustered together}}{\text{Total Events that should be clustered}} \tag{4}$$

$$Precision = \frac{\text{Relevant Event clustered together}}{\text{Total Events clustered together}} \tag{5}$$

The FEC_k algorithm is compared with DBSCAN, which is widely used in many different domains with a various types of data. Moreover, DBSCAN can handle noise and outliers construct clusters of arbitrary shapes and sizes. It has better results than K-means or CLARANS [22]. Its performance has also been improved in [25, 26] but it still has high complexity; $O(n^2)$. Moreover, some of its parameters are not easy to set, such as *eps*; the maximum radius of the neighbourhood and *MinPoint*; the minimum number of points in an Eps-neighbourhood of that point.

As mentioned above, we evaluated our technique on two types of data: simulated data and real-world data. The main reasons of using a network simulator as well as real-world data are

twofold: 1) the ability to simulate any network behaviour as the simulator has been validated through real-world data, and 2) the validation of the technique by stressing it with many different scenarios of network failures. The first type of data is generated by simulating the behaviour of a telecommunication network. The generated data represents the activity of 6 main areas of a network, each of them containing up to 7 cells, during a time window of 2 hours. 4 types of alarms are used with different levels of importance. The simulator takes into account different network parameters, such as maintenance period, life cycle of a hardware component, some statistics about the software failures and their consequences, alarm types and content, etc.

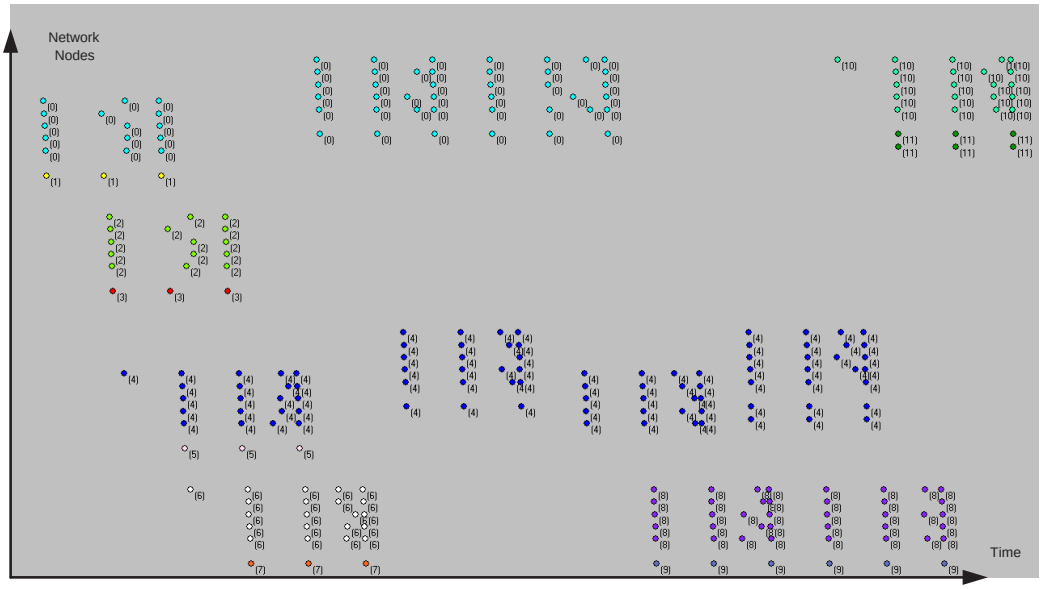

Figure 3. Composition of clusters generated by DBSCAN with simulated data

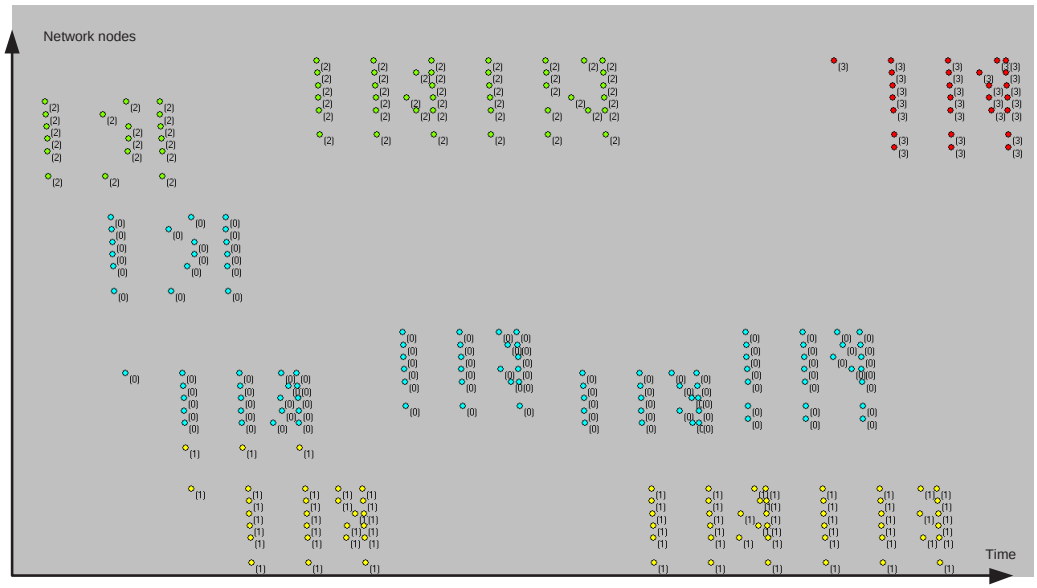

Figure 4. Composition of clusters generated with FEC_k with simulated data

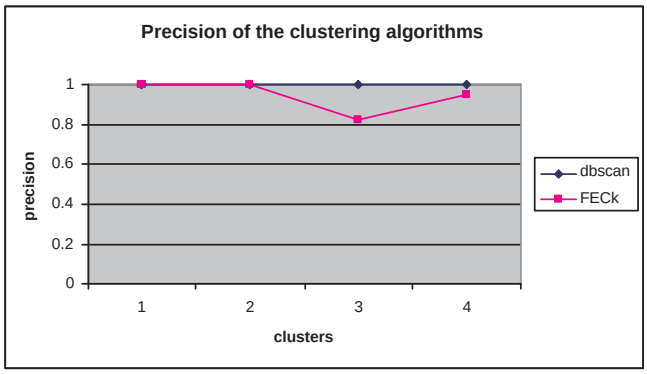

Figure 5. Precision of FECk and DBSCAN on simulated data

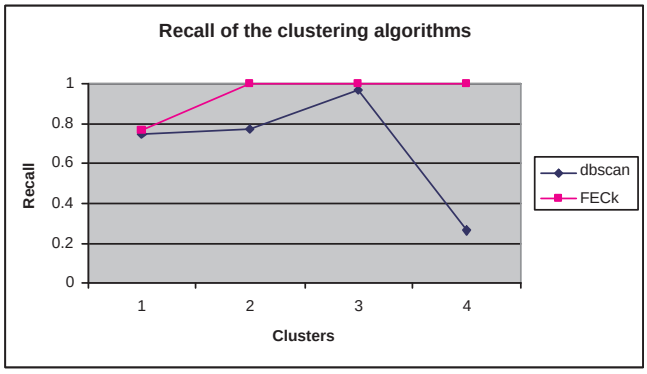

Figure 6. Recall of FECk and DBSCAN on simulated data

Figure 3 shows the clusters composition generated by DBSCAN for one of the simulated telecommunication sub-networks that produced 4800 alarms in a time window of 120 minutes. DBSCAN was setup with *eps* and *minPoint* equal to 1.7 and 2 respectively. From the same Figure, we can see that 12 clusters were identified. We can notice that the clusters $(1, 3, 5, 7)$ were generated with only 3 elements even thought there were some bigger clusters in their neighbourhood. If we look at the composition of these clusters we can see that they are made of telecommunication error messages. In other words they are not relevant for the network operator. They are considered as noise and therefore they are ignored. However, the user intervention is required to check those clusters and make a decision weather they are critical or not.

Figure 4 presents the clusters composition for the same simulated data set using FEC_k. Our algorithm was setup with a degree of confidence of 90%. The algorithm returned 4 clusters and we can notice that some cluster's compositions are quite similar to those of DBSCAN, (See DBSCAN's clusters number (4) and (2) and FEC_k's clusters number (0) and (2) respectively). The DBSCAN's results seem to suffer from over-partitioning due to different densities of events. The FEC_k's results are better as they are insensitive to the density distribution of the events. It is clear that the FEC_k algorithm is more focused on very important clusters which governed by the critical events. This is not the case for the DBSCAN algorithm.

However, we need also to evaluate the compactness and the accuracy of the clusters. We use two functions similar to recall and precision used in information retrieval. The results are shown in Figures 6 and 5. The FEC_k's *recall* is much better than DBSCAN's one. This means that, in the case of FEC_k, all events that should be clustered together are gathered together to form a cluster. DBSCAN doesn't perform very well compared with FEC_k with regard to the *Recall* function. While

the FEC_k's *Precision* within the clusters is still very high, DBSCAN is doing better. The main reason is that the FEC_k algorithm tends to assign non-relevant events to some other clusters. There are two ways of eliminating of noise and outliers. The first way is to include them in some clusters. This way is more suited for noise elimination, as far as they are ignored during the interpretation phase. This is the case here, as the primary event is really the critical representative of its cluster. The second way is to group them into independent clusters, called `bib-clusters`. This is more suitable for the outliers. However, unlike DBSCAN, FEC_k can easily identify them as outliers, because there is no primary events representing these bin-clusters.

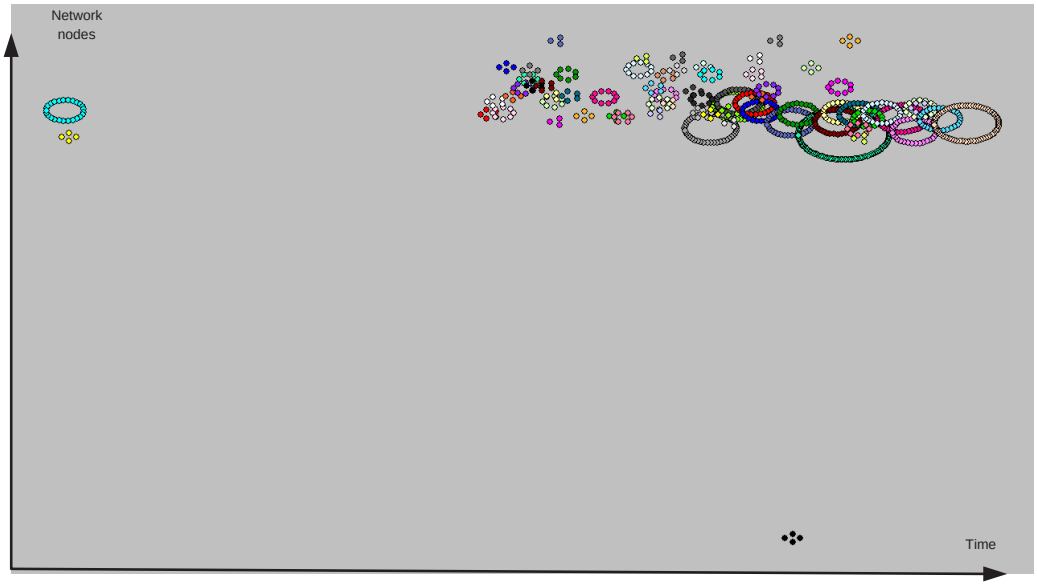

Figure 7. Composition of clusters generated by DBSCAN with real data

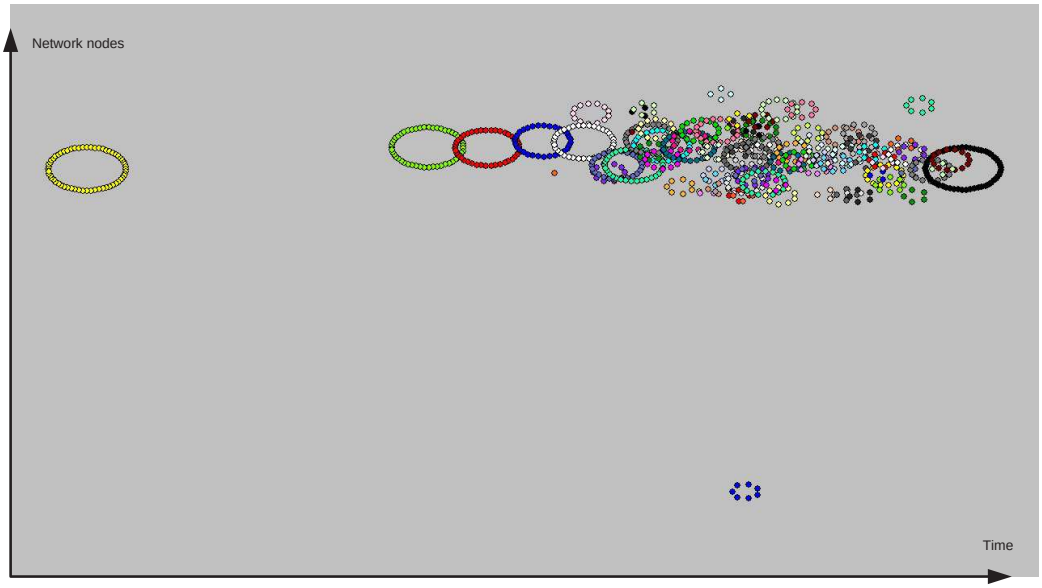

Figure 8. Composition of clusters generated with FEC_k with real data

Figure 7 shows the clusters' composition generated by DBSCAN for real-world dataset. The dataset size is 10080 alarms generated by one sub-network over one day. A circle represents a cluster whose centre has been adjusted using the distance between each event of the cluster. DBSCAN found 69 clusters and the biggest of them is composed by 146 events. Figure 8 shows the clusters generated by FEC_k with the same data. FEC_k identified 95 clusters and the largest on is composed by 60 events. As one can notice, in this case, the FEC_k algorithm produced more clusters than DBSCAN, but they are more balanced. In addition, FEC_k clusters most of the noise and outliers in bin-clusters. These are also returned by the algorithm.

More importantly our technique identifies relevant clusters of events occurring in the early stage of the fault and gives less importance to the events that follows the fault. In other words it identifies the symptoms of the fault and gives less importance to the noise (fault cascade or noisy alarms) generated by the primary event. DBSCAN tends to cluster the events following a fault into large clusters. While it returns small number of clusters compared with FEC_k, they are as relevant as the FEC_k ones. Its' precision and recall drop dramatically in this case and also one can notice that the DBSCAN algorithm does not scale well compared to FEC_k.

5 Conclusion

In this paper we proposed a new efficient algorithm for the alarm correlation problem. The FEC_k algorithm provides the main roots of faults which appeared in the network in the form of clusters with very accuracy. It is fully flexible as it is based on the fuzzy set theory. The algorithm assumes that the data has been pre-processed and consolidated. This is one of the main reason why we have developed a framework for the alarm mining process. This algorithm is evaluated with both simulated and real-world data sets. The results show that FEC_k performs very well and better than DBSCAN. Our algorithm takes into account the data characteristics and the network features. We believe that some faults occur within short time windows around the identified primary events. The other correlated events describe the symptoms and the consequences of this abnormal behaviour. The two-step fuzzy model proved to be very efficient to deal with event correlations and identifying clusters' representatives. For further improvement, we are currently integrating some training skills to this technique in order to recognise and interpret the redundant cluster compositions and also deal with delayed alarms. Because the current version assumes that all primary events are not correlated. However, this can lead to two or more correlated clusters describing the same abnormal behaviour of the network.

References

[1] Gardner, R., Harle, D.: Alarm correlation and network fault resolution using kohonen self-organising map. In: IEEE Global Telecom. Conf. Volume 3., New York, NY, USA (1997) 1398–1402

[2] Bellec, J.H., Kechadi, M.T.: Towards a formal model for the network alarm correlation problem. In: The 6th WSEAS Int'l Conference on Simulation, Modelling and Optimization (SMO'06), Lisbon, Portugal (2006)

[3] Bellec, J.H., Kechadi, M.T., J.Carthy: Performance evaluation of two data mining techniques of network alarms analysis. In: The 2006 Int'l Conference On Data Mining, (DMIN'06), Las Vegas, NV, USA (2006)

[4] Bellec, J.H., Kechadi, M.T., J.Carthy: A new efficient clustering algorithm for network alarm analysis. In: The 17th IASTED Int'l. Conference on Parallel and Distributed Computing and Systems, (PDCS'05), Phoenix, AZ, USA (2005)

[5] Yamanishi, K., Maruyama, Y.: Dynamic syslog mining for network failure monitoring. In: KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, New York, NY, USA, ACM Press (2005) 499–508

[6] Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst. Secur. **6**(4) (2003) 443–471

[7] Li, T., Liang, F., Ma, S., Peng, W.: An integrated framework on mining logs files for computing system management. In: Proc of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining, Chicago, Illinois, USA (2005)

[8] Bertolotto, M., martino, S.D., Ferrucci, F., Kechadi, M.T.: A visualization system for collaborative spatio-temporal data mining. Journal of Geographical Information Science **21**(7) (2007)

[9] M.Bertolotto, Martino, S.D., Ferrucci, F., Kechadi, M.T.: Towards a framework for mining and analysing spatio-temporal datasets. International Journal of Geographical Information Science **21**(8) (2007) 895–906

[10] Compieta, P., Martino, S.D., Bertolotto, M., Ferrucci, F., Kechadi, M.T.: Exploratory spatio-temporal data mining and visualization. Journal of Elsevier Science Special Issue on Human-GIS Interaction **18**(3) (2007)

[11] Meira, D., Nogueira, J.: Modelling a telecommunication network for fault management applications. In: Proc. of NOMS'98. (1998) 723–732

[12] Gopal, R.: Layered model for supporting fault isolation and recovery. In: IEEE/IFIP, Proc. of Network Operation and Management Symposium, Honolulu, Hawaii (2000)

[13] Steinder, M., Sethi, A.: Non-deterministic diagnosis of end-to-end service failures in a multi-layer communication system. In: Proc. of ICCCN'01, Arizona (2001) 374–379

[14] Liu, G., Mok, A., Yang, E.: Composite events for network event correlation. In: IM'99. (1999) 247–260

[15] Yemini, S., Kliger, S., Mozes, E., Yemini, Y., Ohsie, D.: High speed and robust event correlation. IEEE Communications Magazine **34**(5) (1996) 82–90

[16] Wietgrefe, H., Tuchs, K.D., Jobmann, K., Carls, G., Frohlich, P., Nejdl, W., Steinfeld, S.: Using neural networks for alarm correlation in cellular phone networks. In: Proc. of IWANNT. (1997)

[17] Hasan, M., Sugla, B., Viswanathan, R.: A conceptual framework for network management event correlation and filtering systems. In: 6th IEEE/IFIP, Proc. of Network Operation and Management Symposium, Boston, MA, USA (1999)

[18] Chao, C.S., Liu, A.C.: An alarm management framework for automated network fault identification. Computer Communication **27** (2004) 1341–1353

[19] Guha, S., R.Rastogi, K.Shim: Cure: An efficient clustering algorithm for large databases. In: ACM SIGMOD'98, Seattle, WA, USA (1999)

[20] Zhang, T., Ramakrishnan, R., Livny, M.: Birch: An efficient data clustering method for very large databases. In: ACM Intl' Conf. on management of data, 103-114. (1996) 103–114

[21] Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. on Neural Networks **16**(3) (2005) 645–678

[22] Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-base algorithm for discovering clusters in large spatial databases with noises. In: The Int'l Conference on Knowledge discovery and data mining. (1996) 226–231

[23] Zadeh, L.: Fuzzy logic. Computer (1988) 83–93

[24] Ng, R., Han, J.: Efficient and effective clustering methods for spatial data mining. In: The Int'l Very large databases conference. (1994) 144–145

[25] Viswanath, P., Pinkesh, R.: l-dbscan: A fast hybrid density based clustering method. In: 18th IEEE Intl' Conference on Pattern Recognition(ICPR'06). (2006)

[26] Sia, W., Lazarescu, M.: Clustering large dynamic datasets using examplar points. In: The IASTED Conference on Artificial Intelligence and Applications, Innsbruck, Austria (2006)

[27] Batagelj, V., Mrvar, A.: Pajek - Analysis and Visualization of Large Networks. In: Graph Drawing Software. (2003) 77–103

6 Authors

Jacques Henry Bellec is a PhD candidate at the University College Dublin (UCD) in Ireland. His research interests include Data Mining, Pattern Recognition, Information Storage and Retrieval, Clustering Analysis, Clustering Applications, Unsupervised Learning, Fault Management and Alarm Correlation. He has a Post-doctorate certificate preliminary to PhD (DEA), a Master in Computer Science, a Third degree in Computer Science and an Associate degree in Mathematics from the University College Versailles (UVSQ). He is an active reviewer of the WSEAS conferences and journals.

Tahar Kechadi was awarded PhD and a DEA (Diplome d'Etude Approfondie) - Masters degree - in Computer Science from University of Lille 1, France. He joined the School of Computer Science and Informatics, University College Dublin, Ireland in 1999. My research interests span the areas of optimisation techniques, Data mining, heterogeneous distributed systems and Grid computing. He is full member at CERN. He is a visiting professor at the Universities of Artois, France and Oslo, Norway. He is a member of the communication of the ACM Journal and IEEE Computer Society.