Discovering Alarm Correlation Rules for Network Fault Management

Philippe Fournier-Viger[1][0000−0002−7680−9899], Ganghuan He[1], Min Zhou[2], Mourad Nouioua[1,3], and Jiahong Liu[1]

[1] Harbin Institute of Technology (Shenzhen), Shenzhen, China
philfv8@yahoo.com, heganghuan@gmail.com, mouradnouioua@gmail.com, 550565776@qq.com
[2] Huawei Noah's Ark Lab, Shenzhen, China
zhoumin27@huawei.com
[3] University of Bordj Bou Arreridj, Algeria

Abstract. Fault management is critical to telecommunication networks. It consists of detecting, diagnosing, isolating and fixing network problems, a task that is time-consuming. A promising approach to improve fault management is to find patterns revealing the relationships between network alarms, to then only show the most important alarms to network operators. However, a limitation of current algorithms of this type is that they ignore the network topology. But the network topology is important to understand how alarms propagate on a network. This paper addresses this issue by modeling a real-life telecommunication network as a dynamic attributed graph and then extracting correlation patterns between network alarms called Alarm Correlation Rules. Experiments on a large telecommunication network show that interesting patterns are found that can greatly compress the number of alarms presented to network operators, which can reduce network maintenance costs.

Keywords: Fault management · Dynamic graph · Correlation Patterns.

1 Introduction

In today's society, telecommunication networks are key to support personal communications as well as those of businesses and other organizations. To ensure the proper operation of large telecommunication networks, a crucial task is fault management, which consists of detecting, diagnosing, isolating and fixing network problems. The purpose of fault management is to preserve network availability, security, reliability and optimize its performance [1]. However, a key issue with fault management for large and heterogeneous telecommunication networks (e.g. covering cities) is that millions of alarms may be generated by network devices, and that the number of technicians or budget for maintaining a network is limited [3]. Thus, it is easy for technicians to be overloaded with thousands of alarms and being unable to investigate all of them. For example, the telecommunication network of a medium-sized city typically contains multiple device types where some devices may produce more than 300 different alarms. The alarms are recorded by each network device and can be stored centrally and analyzed to support fault management. Moreover, data is also collected about QPIs (Quality Performance Indicators) of each network device over time. For network experts, understanding the relationships between alarms is not easy because faults are often caused by complex interactions between network devices.

To improve fault management, some expert systems were designed that rely on a knowledge base created by hand to find the causes of network problems [2]. But this approach is costly, time consuming, prone to errors and cannot adapt to changes. As an alternative, an emerging approach is to rely on pattern mining techniques to automatically discover relationships between alarms in alarm logs and then to hide (compress) alarms that are correlated with previous alarms [3–7]. It was shown that this can greatly reduce the number of alarms presented to network operators and thus reduce maintenance costs. But such approaches generally represent alarm log data as a sequence of alarms and the network topology is ignored [3–7]. But the topology is important to understand how alarms propagate on a network.

A promising research direction is thus to consider the network topology as a dynamic graph and to extract richer and more complex patterns from it to reveal complex temporal relationships between alarms. Though, several algorithms have been proposed to mine patterns in dynamic graphs, none is specifically designs for alarm analysis [16–19]. To find more complex relationships between alarms based on the network topology, this paper models alarms data as a network (i.e. a dynamic graph) where vertices are devices and edges are communication links. Moreover, alarms are viewed as spreading following the information flow (which depends on the topology) and where QPIs are represented as attributes of network devices. From this representation, this paper proposes to extract a novel type of patterns called Alarm Correlation Rules using a novel correlation measure named ACOR (Alarm CORrelation). An experimental evaluation with real data from a large telecommunication network shows that the proposed rules can provide greater alarm compression than the state-of-the-art AABD system [3].

The paper is organized as follows. Section 2 reviews related work. Section 3 presents the proposed framework. Then, Section 4 describes results obtained for a large scale telecommunication network. Finally, Section 5 draws a conclusion.

2 Related Work

To discover relationships between alarms in telecommunication networks, several studies have applied pattern mining techniques [3–7] such as association rule mining [11] episode mining [8–10] and sequential pattern mining (SPM) [12, 13].

The first system to discover alarm patterns is TASA (Telecommunication Alarm Sequence Analyzer) [5, 6]. It takes as input a sequence of alarms with timestamps and applies an episode mining algorithm to find alarms that frequently appear together within a sliding window. Moreover, TASA offers a separate module that applies association rule mining to find sets of properties that are common to alarm occurrences (while ignoring time). TASA was applied to data from several telecommunication service providers.

Lozonavu et al. [7] proposed a system for mining alarm patterns that first partitions the input alarm sequence into a set of sequences such that alarms having close timestamps are grouped together. Then, a SPM algorithm is applied to find all subsequences of alarms appearing in many of those sequences. Patterns are then used to generate a graph indicating relationships between alarms, where the confidence (conditional probability) that an alarm precedes another is calculated. This visualization can help network operators to understand the relationships between alarms. The system was applied to a 3G mobile network.

Wang et al. [3] proposed a system called AABD (Automatic Alarm Behavior Discovery). This system first filters out invalid alarms (e.g. with missing timestamps) and transient alarms (that appear only for a short time) from the input alarm sequence. Then, the most frequent alarms are identified and the input sequence is partitioned based on these alarms. Then, a SPM [12, 13] algorithm is applied to find frequent alarm sequences. These patterns are then used to generate rules indicating that an alarm may be caused by another alarm to perform alarm compression (reduce the number of alarms presented to network operators). AABD achieved good compression for alarms of a real telecommunication network where it was shown that this approach based on transient alarm detection can reduce the number of alarms presented to operators by more than 84%. But the rule generation process of AABD relies on a knowledge base provided by domain expert, which is time-consuming to create and maintain.

An alarm management system adopting a similar approach was designed by Raúl et al. [4]. It takes as input an alarm sequence with time where alarms have attributes. A modified SPM algorithm was applied to extract sequences of alarms frequently appearing in a sliding-window. Patterns are selected based on three measures that are the support, confidence and lift. The system was applied to data from a large Portugese telecommunication company and patterns were used to reduce the number of alarms presented to the user by up to 70%.

The above pattern mining approaches to study network alarms are useful but handle simple data types, that is mostly discrete sequences where alarms are viewed as events that have some attribute values and timestamps. To extract patterns that consider the network topology and provide different insights, this paper considers a more complex data representation by adding the spatial dimension (the network topology) to the pattern mining process. The network is viewed as a dynamic graph where alarms are spreading along edges (communication links) between vertices (network devices) to find spatio-temporal patterns.

3 The Proposed Framework

This section presents the proposed framework for discovering alarm correlation rules and performing alarm compression. This framework is illustrated in Fig. 1. It consists of three main steps: (1) obtaining and pre-processing alarm and network topology data, (2) extracting alarm correlation rules from it, and (3) utilizing the rules to select alarms to be presented to the user. These three steps are described in details in the next paragraphs.

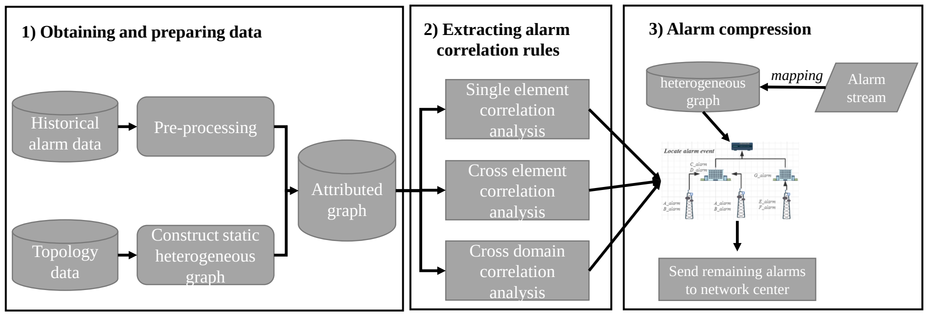
1) Obtaining and preparing data
Historical alarm data
Pre-processing
Topology data
Construct static heterogeneous graph
Attributed graph
2) Extracting alarm correlation rules
Single element correlation analysis
Cross element correlation analysis
Cross domain correlation analysis
3) Alarm compression
heterogeneous graph
mapping
Alarm stream
Send remaining alarms to network center

Fig. 1. The proposed alarm discovery and compression framework

Step 1. Obtaining and preparing the data. In previous studies, historical alarm logs were analyzed to find patterns involving multiple alarms. But most studies represent alarms log data as a sequence of alarms ordered by time. Because these studies ignore the network topology, it may lead to obtaining imprecise results or ignoring some important underlying patterns. In this work, we make the observation that telecommunication alarm data can be naturally modeled as a network (a dynamic graph) in which alarms can spread following the information flow. Thus, we not only consider the historical alarm log but also the network topology. The following paragraphs describes how these two types of data are obtained, pre-processed and then combined.

a) Preparing the historical alarm log. The alarm log format considered in this study is presented in Table 1, and is more or less the same as in prior studies [3]. Each alarm has a name, a source (the device where the alarm was triggered), the domain of the device, an occurrence time and a clear time. For this study, five days of data was obtained from a large telecommunication network in Indonesia, from the 12th to 16th April, 2019. This dataset contains more than six million alarms, categorized into 300 types, triggered by different devices. To ensure privacy, alarm names and sources are not shown in Table 1.

Table 1. Part of an alarm log from an Indonesian telecommunication network

Alarm Name	Domain	Alarm Source	Occurrence Time	Clear Time
Alarm 1	ran-4g	Source 1	2019-04-12 10:40:23	2019-04-12 10:40:29
Alarm 2	microwave	Source 2	2019-04-12 10:40:24	2019-04-12 11:30:44
Alarm 3	ran-2g	Source 3	2019-04-12 10:40:26	2019-04-12 10:40:36
...	...	...	...	...

After obtaining the alarm log, the proposed framework pre-processes the data to filter out some spurious alarms. This is done based on the recommendation of telecommunication network experts and allows to perform a more precise analysis of alarm correlation and to reduce the time required for calculations. First, all alarms that repeatedly appear in a device during a short period of time (five minutes as per the recommendation of domain experts) are combined. Second, some repeatedly occurring alarms whose duration time is very short are filtered out as they are considered uninteresting. Third, alarms that have incomplete information (e.g. an empty alarm source field) are discarded.

b) Building a static heterogeneous graph. After preparing the alarm log, the framework obtains data about the network topology. This data is represented as a connected directed graph where devices are vertices and edges indicate the directions that information flows between devices.

In this study, the network topology was unavailable. Hence, a procedure was designed to extract the topology from logs indicating how information transited through the network. Table 2 depicts part of such log, where the basic component is paths. A path is an ordered list of devices through which some messages have transited. Note that a device may appear in multiple paths.

Table 2. Part of an information flow log

Path Id	Device Name	Device Type	Path Hop
1	Device 1	Router	0
1	Device 2	Microwave	1
1	Device 3	RAN	2
2	Device 1	Router	0
2	Device 4	Microwave	1
2	Device 5	RAN	2
...	...	...	...

By combining paths, a static heterogeneous graph is obtained representing the network topology such as the one shown in Fig. 2 (left). The constructed graph is hierarchical where each node represents a network device. Three types of devices are considered, namely routers, microwave devices and RAN (Radio Access Network) devices (also called NodeB). The information generally flows from routers to microwave devices, and then to RAN devices. The graph generated using the collected data contains 41,143 distinct vertices (devices) and nodes appears to be hierarchically organized into three layers (called domains) as a tree-like structure. However, it should be noted that some nodes are interconnected with others in the Microwave layer. Hence, there are some cycles between microwave nodes and the network must be represented as a graph rather than a tree.

c) Mapping alarms to the network. After obtaining the graph representing the network topology, the proposed framework maps each alarm from the

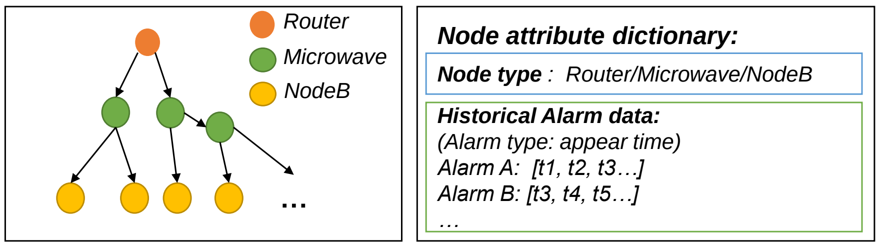

Fig. 2. The recovered network topology (left) with alarm attributes (right)

alarm log to devices of the graph. This is done by matching values for the Device Name field from information flow paths (as in Table 2) to values in the Alarm Source field of the alarm log (as in Table 1). The result is a graph-based data representation where all triggered alarms are encoded in vertex attributes on the topology and where the topology remains fixed. Such structure is depicted in Figure. 2. For a given node, each attribute represents an alarm type and contains the list of alarm occurrences of that type, sorted by time. This graph-based structure indicates how alarms change over time and is a type of dynamic attributed graph. Note that during the mapping process, alarms that are not mapped to any device of the topology are discarded. The next paragraphs explains how interesting correlation patterns are extracted from this data representation.

Step 2. Extracting alarm correlation rules. After the data has been prepared, the proposed framework extracts patterns indicating strong temporal relationships between pairs of alarm types. A natural representation for such relationships is rules of the form $A \rightarrow B$ indicating that if some alarm of type A appears, an alarm of type B is also likely to appear. But finding interesting rules requires to define a measure of the correlation of A and B.

In association rule mining [11, 14], several measures have been proposed to find strong rules such as the *support* (occurrence count of A with B) and the *confidence* (occurrence count of A with B divided by the occurrence count of A). But the support measure is not very suitable for alarm correlation analysis because very frequent alarms are generally unimportant and may even be considered as noise. The *confidence* measure has the drawback that it is very sensitive to the frequency of a rule's consequent (B) in the database. Another traditional measure is the *Lift* measure [14], which is less influenced by the presence of rare items but it is symmetric. In this study, we want an asymmetric measure to help us judge how an alarm influences the other. The lift does not allow to distinguish between the correlation of A with B and that of B with A.

To address the above limitations of the above measures, this paper presents a novel correlation measure named ACOR (Alarm CORrelation) specifically designed for evaluating the correlation between two alarms A and B. Some advantages are that it consider the occurrence frequencies of A and B so that it can minimize the impact of noisy data (some alarms always appear or only appear once or twice). And ACOR amplifies the difference between the associated values. It is worth noticing that it is an asymmetric measure, i.e, $acor_{A2B}$ does not equal $acor_{B2A}$. The measure is given as:

$$acor_{A2B} = \left(\frac{\frac{ABcount}{Acount})}{2 - \frac{ABcount}{Bcount}}\right), \tag{1}$$

where $ABcount$ is the number of time windows where A and B appeared together (e.g. within 5 minutes) which can be interpreted as indicating that A and B may have the same cause and $Acount$ (resp. $Bcount$) is the number of occurrences of alarm A (resp. alarm B) in the log data. The closer a $acor_{A2B}$ value is to the maximum of 1, the higher the correlation between the two alarms is.

Besides, it can be observed that the correlation measure is designed to not be strict about the order of occurrences between two alarms A and B, as long as they occur together closely enough (whithin a time window). The reason for not requiring a strict order between A and B is that clocks of network devices are not perfectly synchronized. As a result, some event may appear before another event in the alarm log although it actually appeared after.

Another contribution of this work is to not only find rules about alarms within a single device (single device rules) but also between devices from the same domain (cross device rules) and between devices of different domains (cross domain rules). This is useful because a telecommunication network is typically hierarchical, and devices within each layer (domain) behave quite differently. Nodes from different domains also have completely different types of alarms and communication link between nodes are also determined by the domains containing these nodes. Discovering cross device and cross domain rules allows to go beyond simple correlations occurring within a single device to find patterns applicable in other scenarios. This was not done in previous studies as the network topology was ignored.

To find correlation rules between alarms, a data mining algorithm is applied to the previous graph structure. As mentioned, this paper considers three scenarios for alarm correlation analysis: single device rules, cross device rules and cross domain rules. To find these rules, the correlation between all pairs of alarms is calculated according to the $acor_{A2B}$ formula. Algorithm 1 shows the pseudo code for calculating the correlation of a single device rule. It takes as input the graph data structure previously built, two alarm types A and B, and returns the correlation of $A \rightarrow B$. In the pseudocode, the notation $len(node.alarm\_A)$ represents the number of alarms of type A that have occurred in a given device called $node$, and $node.alarm\_A[i]$ refers to the i-th alarm occurrence of type A occurring in the device $node$. The algorithm can be easily extended to identify strongly correlated cross device and cross domain rules. The only difference between these different scenarios is that alarms must be in different positions in the network when calculating the correlation. Finally, the rules are ranked by decreasing order of correlation. The assumption is that rules having a high correlation are more interesting. The rules can be analyzed by an expert or used for alarm compression as it will be explained in the next subsection.

Algorithm 1: Calculating the correlation of a single device rule

```
input : a dynamic attributed graph $\mathcal{G}$,
        two alarm types $A$ and $B$
output: Correlation value of A to B
1  Initialize $A\_count \leftarrow 0, B\_count \leftarrow 0, AB\_count \leftarrow 0$
2  foreach $node \in \mathcal{G}$ do
3      $A\_count \leftarrow A\_count + len(node.alarm\_A)$
4      $B\_count \leftarrow B\_count + len(node.alarm\_B)$
5      Initialize $i \leftarrow 0, j \leftarrow 0$
6      while $i < len(node.alarm\_A)$ and $j < len(node.alarm\_B$ do
7          if $node.alarm\_A[i], node.alarm\_B[j]$ appear together then
8              $AB\_count \leftarrow AB\_count + 1$
9              $i \leftarrow i + 1$
10             $j \leftarrow j + 1$
11         end
12         if $node.alarm\_A[i]$ appears before $node.alarm\_B[j]$ then
13             $i \leftarrow i + 1$
14         end
15         if node.alarm_B[j] appears before node.alarm_A[i] then
16             $j \leftarrow j + 1$
17         end
18     end
19 end
20 $acor \leftarrow (AB\_count/A\_count)/(2 - (AB\_count/B\_count))$
21 return cor
```

Step 3. Compressing alarms using the alarm correlation rules. After extracting alarm correlation rules, the framework utilizes the discovered alarm correlation rules for alarm compression. This is done in two steps.

a) Aggregating rules and inferring the cause of an alarm. First, the top-k alarm correlation rules are selected where k is a parameter that is set by the user. This is to avoid having to process a very large number of rules in the subsequent step.

For single device correlation analysis, if a more compact representation is required, an inference graph can be created between alarms of different domains. There will be a small connected subgraph in the inference graph where vertices are alarms and edges are the relations between alarms. Then we can get a number of connected subgraphs that is independent alarm sets. Note that the proposed method uses the property that the value of the correlation is not symmetric to delete edges in the inference graph, and simply obtain each P alarm through the inference graph. For cross device and domain correlation analysis, we directly infer the direction of the information flow in the network to get P alarms.

b) Filtering alarms in real-time. Then, the framework applies alarm correlation rules selected in the previous step to filter alarms. But a challenge is that the network center in charge of the telecommunication network receive a constant flow of alarms. To be able to filter alarms in real-time, the proposed framework is adapted to uses a sliding window. For each window, alarms of that window are mapped to the graph representing the network topology to create an attributed graph. Then, the framework respectively performs pre-processing filtering, cross domain compression, cross device compression and single device compression using the rules obtained by the knowledge discovery process. Lastly, the remaining alarms from the network are reported to the network management center and some technicians will be dispatched to check and fix the nodes (devices) having alarms.

4 Experimental Evaluation

To evaluate the proposed framework, two experiments were done using real alarm data collected from an Indonesian telecommunication network (described in Section 3). Results where compared with rules found by the state-of-the-art AABD [3] system, obtained from its authors.

Rule quality. The first experiment was carried out to verify the quality of the alarm correlation rules extracted by the proposed framework. For this purpose a comparison was made with the 135 rules found by the AABD system [3] to see if rules found by AABD could be rediscovered and if many other rules with a similar or higher correlation could be found. The rules found by AABD are used as baseline as they have been verified as valid by domain experts. Both approaches were applied using the same time window of 5 minutes, suggested by domain experts. Let A and B denote the sets of rules found by AABD and the proposed framework, respectively. The coverage ratio was calculated, which is defined as $coverage = |A \cap B|/|B|$. A high coverage ratio indicates that many rules of the proposed framework are valid (as A was validated by experts). However, it should be noted that this measure does not give a full picture as there may exist valid rules not found by AABD.

To select good rules, a minimum correlation threshold was applied in the proposed framework. As this parameter is set lower, more rules may be found, and then the coverage ratio may increase but the accuracy of rules may decrease. It is thus important to choose a suitable value for this parameter that is not too low to avoid finding many spurious rules. To choose a suitable value, we applied the empirical "elbow method" approach for setting a parameter, which is commonly used in data mining and machine learning [15]. It consists of drawing a chart representing the impact of a parameter on a measure to find the point where further increasing or decreasing the parameter would result in a huge change for that measure. In this study, we varied the minimum correlation threshold and noted the number of rules found for each value to draw a chart representing the frequency distribution of rules w.r.t correlation (shown in Fig. [15]). We then observed that a large increase in the number of rules occurs for correlation values below 0.135. Assuming that those could be spurious rules, we set the minimum correlation to 0.135. For this parameter value, about 500 alarm correlation rules were discovered including 113 found by AABD. Thus, in this case, the coverage ratio is 113/135 = 84%. It was observed that many rules not found by the AABD system were discovered that have similar or higher correlation values than rules found by AABD. This is interesting as they are new rules exclusively discovered by the proposed framework that may be valid rules. Rules were presented to a domain expert who found that the majority of the new rules are interesting.

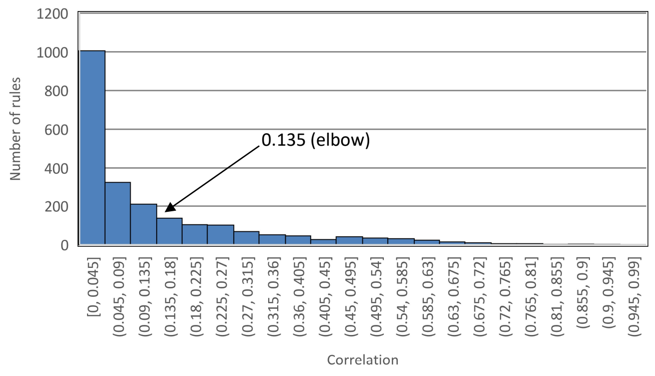

Fig. 3. The correlation distribution of rules

Alarm compression rate. The second experiment aimed at evaluating the number of alarms that could be compressed (removed) using the discovered alarm correlation rules. The original number of alarms and the number of remaining alarms after applying each compression procedure is shown in Table 3. After keeping only the alarms triggered by devices from the reconstructed topology, 4,481,273 alarms were kept from the original 6,199,650 ones. Then, preprocessing was applied, which further reduced that number to 992,966. Then, alarms were compressed using cross domain, cross device, and single device alarm correlation rules, respectively. In the end, 590,307 alarms remained, that is 9.5% of the original alarms. The **final compression rate** obtained by the system is thus $\frac{590{,}307}{4{,}481{,}273} = 87.9\%$. This is a big reduction that can greatly reduce the work of network operators.

Table 3. Remaining alarms count after applying each compression procedure

Compression procedure	Remaining alarms count
Original alarms count	6,199,650 (100%)
Available alarms on topology	4,481,273 (72.2%)
After pre-processing	992,966 (16.0%)
After cross domain compression	874,770 (14.1%)
After cross device compression	756,316 (12.2%)
After single device compression of microwave domain	634,855 (10.2%)
After single device compression of ran domain compression	590,307 (9.5 %)

To put this into perspective, the compression obtained using the alarm correlation rules was compared with that obtained using rules found by AABD. While the proposed framework can find three main types of rules (cross domain, cross device and single device), AABD only finds single devices rules in the RAN domain. Thus, a comparison of the obtained compression was made using only this type of rules. Using rules found by AABD, 39,603 alarms were removed, while 44,548 were removed using the proposed framework. Thus, the proposed method allowed to remove 12.5% more rules than AABD for the scenario of single device compression. If the other types of rules found by the proposed framework are also used, a much greater compression can be obtained. For example, if cross domain rules are also used, 162,744 alarms are removed by the proposed framework, that is 123,141 more than AABD. And if both cross domain and cross device rules are utilized as well as single device rules from all domains (not just RAN), 284,463 more alarms are removed compared to AABD. Note that AABD could also be used to find rules in other domains but such rules were not provided.

5 Conclusion

To find interesting correlations between triggered alarms in telecommunication networks, we modeled a network as a dynamic attributed graph where alarms are viewed as device (node) attributes. A framework was designed to extract correlation rules for a single device, between different devices and across different domains. For this, a novel correlation measure named ACOR (Alarm CORrelation) was designed. By considering the network topology, the rules can reveal interesting relationships between alarms not found by prior approaches. The solution was applied to data from a large telecommunication network. Interesting patterns were discovered and it was found that the patterns can provide greater alarm compression than the state-of-the-art AABD system [3]. This reduces the number of alarms to be analysed by network operators and thus the costs of network maintenance. In future work, we plan to extract more complex graph-based patterns to reveal other types of interesting information from network alarm logs and designing distributed algorithms for processing very large alarm logs.

References

1. Dusia, A., Sethi, A.S.: Recent advances in fault localization in computer networks. IEEE Communications Surveys and Tutorials **18**(4), 3030–3051 (2016)
2. Ding, J., Kramer, B., Xu, S., Chen, H., Bai, Y.: Predictive faultmanagement in the dynamic environment of ip networks. In: 2004 IEEE Intern. Workshop on IP Operations and Management, pp. 233–239 (2004)
3. Wang, J., He, C., Liu, Y., Tian, G., Peng, I., Xing, J.,Ruang, X., Xie, H., Wang, F. L. Efficient alarm behavior analytics for telecom networks. Information Sciences, **402**, 1–14 (2017)
4. Costa, R., Cachulo, N., Cortez, P.: An intelligent alarm management system for large-scale telecommunication companies. In: Portuguese Conf. on Artificial Intelligence, pp. 386–399. Springer (2009)

5. Klemettinen, M., Mannila, H., Toivonen, H.: Rule discovery in telecommunication alarm data. Journal of Network and Systems Management **7**(4), 395–423 (1999)
6. Hatonen, K., Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H.: Tasa: Telecommunication alarm sequence analyzer or how to enjoy faults in your network. In: Proceedings of NOMS'96-IEEE Network Operations and Management Symposium, vol. 2, pp. 520–529. IEEE (1996)
7. Lozonavu, M., Vlachou-Konchylaki, M., Huang, V.: Relation discovery of mobile network alarms with sequential pattern mining. In: 2017 Intern. Conf. on Computing, Networking and Communications (ICNC), pp. 363–367. IEEE (2017)
8. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovering frequent episodes in sequences. In: Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining (1995)
9. Ao, X., Shi, H., Wang, J., Zuo, L., Li, H., He, Q.: Large-scale frequent episode mining from complex event sequences with hierarchies. ACM Transactions on Intelligent Systems and Technology (TIST) **10**(4), 1–26 (2019)
10. Fournier-Viger, P., Wang, Y., Yang, P., Lin, J. C.-W., Yun, U.: TKE: Mining Top-K Frequent Episodes. In: Proc. 33rd Intern. Conf. on Industrial, Engineering and Other Applications of Applied Intelligent Systems, pp. 832-845 (2020)
11. Luna, J.M., Fournier-Viger, P., Ventura, S.: Frequent itemset mining: A 25 years review. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **9**(6), e1329 (2019)
12. Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Mining sequential patterns by pattern-growth: the PrefixSpan approach. IEEE Trans. Knowledge Data Engineering **16**(11), 1424–1440 (2004)
13. Truong, T., Duong, H., Le, B., Fournier-Viger, P.: Fmaxclohusm: An efficient algorithm for mining frequent closed and maximal high utility sequences. Engineering Applications of Artificial Intelligence **85**, 1–20 (2019)
14. Tsang, S., Koh, Y.S., Dobbie, G.: Finding interesting rare association rules using rare pattern tree. Transactions on Large-Scale Data-and Knowledge-Centered Systems **8**, 157–173 (2013)
15. Liu, F. and Deng, Y.: Determine the number of unknown targets in Open World based on Elbow method. IEEE Transactions on Fuzzy Systems (2020)
16. Kaytoue, M., Pitarch, Y., Plantevit, M. and Robardet, C.: Triggering patterns of topology changes in dynamic graphs. IEEE/ACM Intern. Conf. on Advances in Social Networks Analysis and Mining (ASONAM 2014) (2014).
17. Fournier-Viger, P., He, G., Cheng, C., Li, J., Zhou, M., Lin, J. C. W., and Yun, U.: A survey of pattern mining in dynamic graphs. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, pp. e1372 (2020).
18. Fournier-Viger, P., Cheng, C., Cheng, Z., Lin, J.C.W. and Selmaoui-Folcher, N.: Finding Strongly Correlated Trends in Dynamic Attributed Graphs. Intern. Conf. on Big Data Analytics and Knowledge Discovery, pp. 250–265 (2019).
19. Desmier, É., Plantevit, M., Robardet, C. and Boulicaut, J.F.: Granularity of co-evolution patterns in dynamic attributed graphs. Intern. Symposium on Intelligent Data Analysis ;pp. 84–95 (2014).