Alarm Sequence Rule Mining Extended With A Time Confidence Parameter

[ξ]Ömer Faruk Çelebi, [‡]Engin Zeydan, [&]Ismail Ari,
[§]Ömer İleri, and [‡]Salih Ergüt ⋆

[ξ] EMC Turkey, İstanbul, Turkey, e-mail: omerf.celebi@gmail.com,
[‡]AveaLabs, İstanbul, Turkey, email: {engin.zeydan, salih.ergut}@avea.com.tr,
[&]Department of Computer Engineering, Özyegin University, Istanbul, Turkey, email: ismail.ari@ozyegin.edu.tr,
[§]Cypress Semiconductor, Istanbul Turkey, email: omer.ileri@gmail.com

Abstract. Most mobile telecommunication operators receive an overwhelming number of alarms in their networks. Network support specialists are faced with the challenge of picking the most important alarms in advance that can cause severe damages to the system or disrupt the service. A system that can discover alarm correlations and alarm rules then notify network administrators can significantly increase the efficiency of Network Operation Centers (NOC) of these mobile operators. This paper provides a new alarm correlation, rule discovery, and significant rule selection technique based on analysis of real data collected from a mobile telecom operator. We present a method based on sequential rule mining algorithm with an additional parameter called *time-confidence*. The *time-confident* rules found by this method are processed more efficiently in real-time Complex Event Processing (CEP) systems that require exact time-window values during monitoring. Furthermore, compared to traditional sequential rule mining, our proposed method adds another support dimension to eliminate meaningless rules that appear due to wrong settings of minimum support-confidence thresholds with respect to the nature of data.

Keywords: time confidence, alarm correlation, sequential rule mining, complex event processing, telecom network operation.

1 Introduction

Large telecommunication networks have thousands of devices/ network equipments that generate different sets of alarms. Due to the large volume of alarms, network operators may overlook or misinterpret some of the important alarm data. Additional Operational and Capital Expenditures (OpEx,CapEx) by the operators are incurred when the network support administrators spend all of

⋆ This work has been sponsored by AveaLabs, TUBITAK TEYDEB 1501 no. 3130411 and TUBITAK 3501 no. 109E194 projects

their time for analyzing and interpreting the daily alarm information. In addition, faults that can interfere with routine services offered by the network operator decrease quality of service. This could decrement the competitive advantage of an operator and lead to customer churn. In order to circumvent this, the network management systems should automatically apply efficient alarm filtering, semantic aggregation and/or rule discovery procedures to reduce the high numbers and varying types of daily alarms that are received in Network Operation Centers (NOC). In this context, discovering alarm correlations and extracting the most meaningful rules has a key importance.

The motivation for our work was the lack of an accurate and reliable alarm rule discovery and reduction framework. Real time alarm tracking or event sequence detection systems [1] use a fixed window size which may not be appropriate in real settings. Due to vast amount of alarms and tickets that goes beyond manual management capabilities, automatic alarm rule discovery is of vital importance. In this paper, we try to answer the following questions:

1. How can discovered alarm sequences be reduced effectively?
2. How should the constructed alarm rules be registered into a real-time warning system, so that the system works more efficiently?

We learned that in order to answer both of the questions above, the time dimension must be taken into consideration. We observed that the time difference between two alarm events can help operators to predict and take appropriate actions before the subsequent alarm event occurs. Accordingly, obtaining more time-related knowledge from the observed sequences can reveal a better understanding of the internal structure of alarm events.

2 Related Work

In the analysis of alarm flows in telecommunication networks, there are different approaches for dealing with the alarm correlation problem [2] [3]. Some of the approaches focus more on implementing the alarm correlation engine [4], while others focus on alarm modeling and validation scenarios [3].

There is also numerous prior work on temporal data mining that observe frequent patterns in a sequence database (see [5] for a survey). Some of these papers have considered only Top-K event sequence detection in an alarm database [6] [7]. While Top-K analysis provides a good descriptive base for the past, it usually does not have any predictive power even for the near future. On the predictive front, algorithms such as GSP [6] and WINEPI [7] were the first to apply Apriori algorithm [8] to find sequential association rules temporally. These algorithms require a user-defined and fixed sliding-time window duration to traverse the data. In this research, we find and demonstrate that one fixed window size cannot be used to efficiently detect all (i.e. different) sequential rules and a per-rule window size is ideal.

Most of the sequential data mining methods proposed so far are based on discovering order relations between events [6]. These methods are based on finding the frequencies of event sequences and generating the rule candidates against

the database [9]. Some have concentrated on prediction problems for sequential rule mining on several different application domains [9] [10]. However, these algorithms do not focus on time-interval differences between events (item sets) in a rule and generate many uninteresting rules [11]. Wu. *etal.* have defined an *urgent window* parameter where a fixed time range interval selected by users will ensure that the events happening during this interval will become a valid rule [12]. However, this paper does not discover the potential time interval patterns among events for setting the urgent window size.

3 Event Sequence Model

In this paper, we consider the problem of sequential rule mining for correlated sequences. We both formally define the problem and give a clarifying example in this section. Let a *sequence database SeqDB* be the set of event sequences denoted by $SeqDB = \{c_1, c_2, ..., c_k\}$ where $c_i =< seq\_id, s_i >$ represents a *transaction*, *seq_id* is the identifier of a transaction, each $s_i = \{X_1, X_2, ..., X_m\}$ is a subset of event occurrence itemsets from $I = \{i_1, i_2, ..., i_n\}$ where $X_1, X_2, ..., X_m \subseteq I$ [9]. In the context of network alarm operations, note that each item corresponds to a specific alarm. The length of the sequence set $SeqDB$ is denoted by $|SeqDB|$. Every itemset X_k has a special attribute called *timestamp*, which records the time when the item occurred. Note in Figure 1, each row corresponds to a transaction whereas each column represents a specific itemset that occurs at the indicated time instance. Note also that a specific itemset X_i can consist of multiple items each of which corresponds to a specific alarm.

A sequential rule r between X_i and X_j implies that every occurrence of itemset X_i is followed by the occurrence of itemset X_j, such that $X_i \cap X_j = \emptyset$ and X_i and X_j are non-empty. Note that the items inside X_i and X_j need not have a specific order internally, i.e. they can be unordered among themselves. Let R be the set of all such relations r extracted from all analyzed transactions.

In sequential rule mining [7] [8] [10], two different measures can be defined : the *sequential support* (denoted here as *seqSup* in short), given as

$$seqSup(X_i \Rightarrow X_j) = \frac{sup(X_i \blacksquare X_j)}{|SeqDB|},$$

where $sup(X_i \blacksquare X_j)$ denotes the number of occurrences where all items of X_i appear before all the items of X_j [13].

Similarly, *sequential confidence* (denoted here as *seqConf*) given as

$$seqConf(X_i \Rightarrow X_j) = \frac{sup(X_i \blacksquare X_j)}{sup(X_i)}$$

where $sup(X_i)$ denotes the support of X_i, *i.e.* the number of occurrences of X_i in the transaction database. Figure 1 shows the corresponding *seqSup* and *seqConf* for the transactions listed. Note that *seqSup* indicates a measure of the relative frequency of the occurrence of rule r whereas *seqConf* is an indicator of

the expectance with regards to the occurrence of X_j given X_i has already been realized.

In addition to the above, this paper introduces a new metric called *sequential time confidence* (denoted here as *time_conf*) which is defined as:

$$time\_conf(X_i \Rightarrow X_j; TI(r)) = \frac{sup(X_i \blacksquare X_j; TI(r))}{sup(X_i \blacksquare X_j)} \quad (1)$$

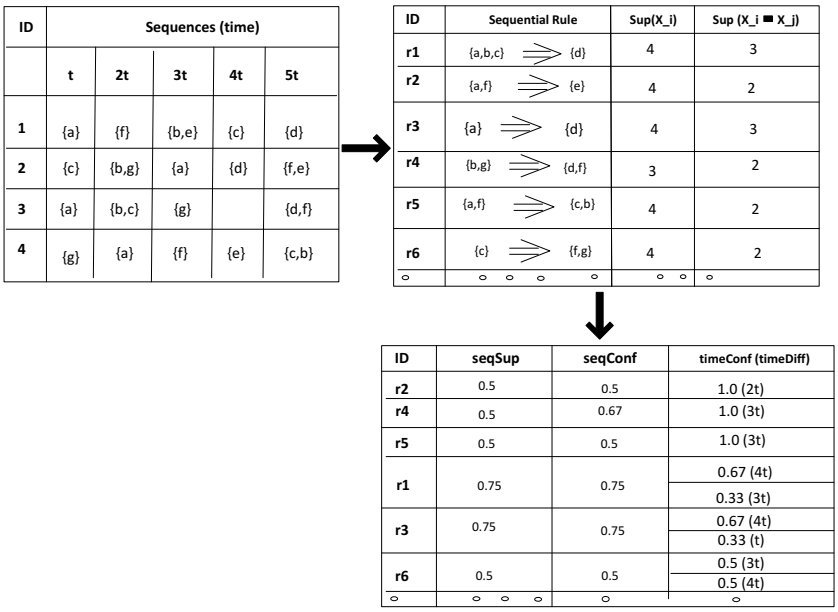
where $TI(r) = \{\{T_i, a_i\} | i \in L\}$ is the set of time intervals T_i, which denotes a specific time interval length between X_i and X_j, and a_i that denotes the number of times that X_j is followed by X_i after exactly T_i units of time. Here, L is the set of distinct T_i. Note that in case of chain occurrence rule $\{X_i, X_p\} \Rightarrow \{X_j, X_k\}$, time interval T_i is taken to be the duration that starts with the realization of X_i or X_p whichever comes first and ends with the occurrence of X_j or X_k, whichever comes last. Note also that time confidence is actually a distribution over the time intervals between the realization of related itemsets X_i and X_j.

A meaningful set of rules would be achieved by selecting the rules that have *seqSup*, *seqConf* as well as *time_conf* values greater than user-defined $minSeqSup$, $minSeqConf$ and $minSeqTimeConf$ thresholds, respectively. A rule or pattern that has support greater than a $minSeqSup$ is considered as *frequent*, a rule with a confidence greater than $minSeqConf$ is *confident* and a rule with a time-confidence higher than $minSeqTimeConf$ is considered to be *time_confident* or *significant*.

As an example, consider the sequence database shown in the left part of Figure 1. The right part of Figure 1 illustrates some of the significant rules found in this database for $minSeqSup = 0.5$, $minSeqConf = 0.5$ and $minSeqTimeConf = 0.33$. Consider the rule $r4$ where $T(r4) = \{\{3t, 2\}\}$, which means that after the events $\{b\}$ and $\{g\}$ occur in any time order, events $\{d\}$ and $\{f\}$ will occur in any order within time intervals $3t$ with number of occurrences of 2. Its sequential support is $sup(\{\{b\}, \{g\}\} \blacksquare \{\{d\}, \{f\}\})/|SeqDB| = 2/4 = 0.5$, its sequential confidence is $sup(\{\{b\}, \{g\}\} \blacksquare \{\{d\}, \{f\}\})/sup(X_i) = 2/2 = 1$, whereas its sequential time-confidence is $sup(\{\{b\}, \{g\}\} \blacksquare \{\{d\}, \{f\}\}; TI(r))/sup(X_i \blacksquare X_j) = 2/2 = 1$ for $3t$ time interval length. Because those values are no less than $minSeqSup$, $minSeqConf$ and $minSeqTimeConf$, the rule will be significant.

4 A Sequential alarm correlation method with time−confidence

The problem of proposed method is to find the complete set of alarm rules that satisfy a given minimum support, confidence and time-confidence thresholds in the alarm transaction database. The time-confidence of a rule is basically calculated over the distribution of the time intervals between the alarm events in a sequential association rule. Therefore, the proposed method extracts only significant rules that also have a time-interval based validity, which is a subset of all rules found by the traditional sequential rule mining algorithm that only uses



ID	Sequences (time)				
	t	2t	3t	4t	5t
1	{a}	{f}	{b,e}	{c}	{d}
2	{c}	{b,g}	{a}	{d}	{f,e}
3	{a}	{b,c}	{g}		{d,f}
4	{g}	{a}	{f}	{e}	{c,b}

ID	Sequential Rule	Sup(X_i)	Sup (X_i ⇒ X_j)
r1	{a,b,c} ⇒ {d}	4	3
r2	{a,f} ⇒ {e}	4	2
r3	{a} ⇒ {d}	4	3
r4	{b,g} ⇒ {d,f}	3	2
r5	{a,f} ⇒ {c,b}	4	2
r6	{c} ⇒ {f,g}	4	2

ID	seqSup	seqConf	timeConf (timeDiff)
r2	0.5	0.5	1.0 (2t)
r4	0.5	0.67	1.0 (3t)
r5	0.5	0.5	1.0 (3t)
r1	0.75	0.75	0.67 (4t)
			0.33 (3t)
r3	0.75	0.75	0.67 (4t)
			0.33 (t)
r6	0.5	0.5	0.5 (3t)
			0.5 (4t)

Fig. 1. An example of an event sequence database at different times when $|SeqDB| = 4$ (left) and the obtained sequential rules (right) for $minSeqSup \geq 0.5$, $minSeqConf \geq 0.5$ and $minSeqTimeConf \geq 0.33$

minimum support and confidence values. These significant rules with time confidence can now be placed into the Complex Event Processing (CEP) engines [14] for accurate, reliable and efficient real-time alarm management.

The procedure for the proposed method is explained in Algorithm 1, input of which is an event sequence database and the output are significant (or time-confident) rules. In Algorithm 1, sequential rule mining algorithms mentioned in step 1 may include RuleGrowth [9], GSP [6], PREFIXSPAN [15], SPADE [16]; we used the RuleGrowth in this paper. Note also that at step 4 of Algorithm 1, T_{time_conf} value is obtained for each rule. This T_{time_conf} value can be used as a window time $T_{registered}$ for CEP systems. Table 1 shows a sample CEP query where the window is statically set to X. The result of this algorithm will be to set/update that parameter for every event sequence discovery. Suppose that if $T_{registered} > T_{time\_conf}$ is selected, then more data will have to be stored in memory and if $T_{registered} < T_{time\_conf}$ is selected, then most of the rules will be missed by the CEP engine. The proposed method also reduces the candidate rule generations compared to traditional sequential rule mining algorithms as seen in step 3 of Algorithm 1. The time complexity of the algorithm is dominated by the calculation of step 2 where the time difference of each item on the left and right side of the rule are compared because sets X_i and X_j do not guarantee time based

Algorithm 1 Proposed method

Inputs:
$SeqDB$: Event Sequence Database;
$min\_sup, min\_conf, min\_time\_conf$;
Outputs: Significant Alarm Rules contained in $SeqDB$
Method:

1. Scan the database $SeqDB$ to discover the sequential rules $r \in R$ using a sequential rule mining algorithm based on min_sup and min_conf.
2. Calculate the event time interval differences of all sequential alarm rules with their corresponding number of occurrences and record them in $TI(r) = \{\{T_i, a_i\} | i \in L\}, \ \forall r \ \in R$.
3. Calculate the $time\_conf$ of each item in $TI(r)$ for rules $r \in R$ using Equ.1 and remove each item in $TI(r)$ if $time\_conf < min\_time\_conf$. If $TI(r) = \phi$, then exclude r from R.
4. Output remaining rules in R with their corresponding $TI(r) = \{\{T_i, a_i\} | i \in L\}$ values and select $T_{time\_conf} = \arg\underset{T_i}{\text{maximum}} \{a_i : \{\{T_i, a_i\}\}, i \in L\}$

ordering internally. Our future work will include performance optimizations in this front.

Table 1. A sample CEP query

SELECT Alarm Instance (for Rx)
FROM NOC Database
WHERE Rx: e.g. { A,B } ⇒ C
WITHIN X sec.

5 Simulation Results

We implemented our proposed method on top of the RuleGrowth, which is a sequential association rule mining algorithm [9]. To evaluate our method, we used a historical alarm database from operational data of a large mobile telecom operator with nearly 15 million customers as of 2014. This operator's NOC receives up to one million alarms per day that are generated and transmitted from different network elements and IT infrastructure systems. For our experiments, we have used only the core network alarm set, which consists of 50 MGW (media gateway), 9 MSC (mobile switching center), 17 NE-3G (3G Network Equipment, UMTS) and 6 OMC (Operation & Maintenance Center) elements. Table 2 shows

some of the most common received alarms from MGW (Media Gateway) in a large mobile operator's network.

We show a sample of alarm database on 25^{th} of May 2012 here with a total of $34,584$ core network alarm events and observe the set of alarm rules in this data for only MGWs. Note also that all the alarm logs generated by each media gateway are considered as one transaction c_i for our simulations.

Table 2. Sample Alarm Messages

ALARM INDICATION SIGNAL (AIS) RECEIVED
BATTERY LOW
BCCH IS NOT AT PREFERRED BCCH TRX
BTS FAULTY
BTS OPERATION DEGRADED
CABINET OPEN
DC POWER
FAULT RATE MONITORING
LINK SET UNAVAILABLE
PCM FAILURE
ROUTE SET UNAVAILABLE
RTCP SUPERVISION FAILURE
SIGNALLING LINK OUT OF SERVICE

We set values $min\_supp = 12$ (per count), $min\_conf = 0.8$ and $min\_time\_confidence = 0.9$, without loss of generality. Under this setup, without $min\_time\_confidence$ value RuleGrowth algorithm finds 25 rules, whereas this number was reduced by our method to only 9 significant rules to be tracked carefully by the CEP system. We verified the validity of the 9 rules suggested by our system and the interval for their co-occurrences (i.e. $time\_confidence$) by running several interviews with our network support field experts. These time-confident rules found automatically, indeed demonstrated the real system behavior. Yet, our contribution would mean 64% reduction in the number of rules to be examined by experts, thus same amount of savings in time. This finding answers our first question in Section I, about reducing the alarm sequence sizes effectively. Figure 2 shows the time difference histogram between the left and right sides of one of the 9 alarm rules ({AIS RECEIVED, LINK SET UNAVAILABLE, ROUTE SET UNAVAILABLE,SIGNALLING LINK OUT OF SERVICE} ==> {FAULT RATE MONITORING}). This rule shows that when links and routes go down, the performance monitoring systems as well as their alarms will also go off. The x-axis is time difference values in seconds, the y-axis is the number of occurrences of this rule for this time interval. In this figure, the longest green column represent the time interval value above the $min\_time\_confidence$ value and red columns represent the ones below this threshold. As seen in Figure 2, in 15 transactions an alarm type FAULT RATE MONITORING is observed exactly after $T_{time\_conf} = 1263$ seconds (about 22 minutes) after the first element in the alarm

set consisting of {AIS RECEIVED, LINK SET UNAVAILABLE, ROUTE SET UNAVAILABLE, SIGNALLING LINK OUT OF SERVICE} is observed. Only one rule with time difference value $T_{registered} = 1263$ seconds will be registered as an optimal window time interval for the CEP system. Thus, event sequences beyond this value will be removed saving the memory and CPU. Hence, we answer our second question in Section I, about efficiency of the real-time queries..

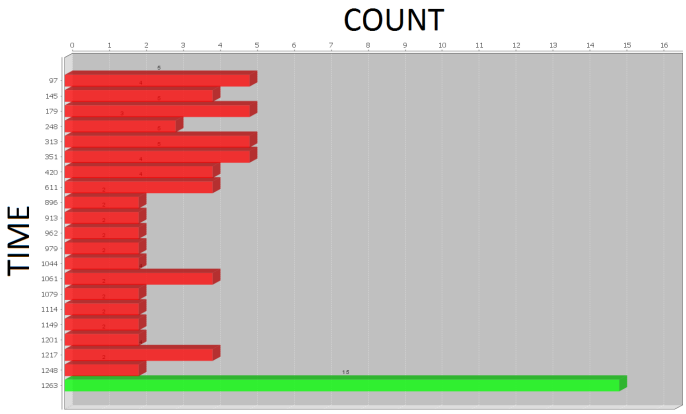


Fig. 2. Time-confidence occurrence histogram for alarm rule ({AIS RECEIVED, LINK SET UNAVAILABLE, ROUTE SET UNAVAILABLE, SIGNALLING LINK OUT OF SERVICE} ==> {FAULT RATE MONITORING})

Our experiments with other days' datasets for the same equipments gave us very similar results and savings in generated rule counts. Therefore we omit those details here for brevity.

6 Conclusions

In this paper, we presented a new method for mining sequential association rules effectively and used this method to address a real problem in mobile network operators. The proposed method has an extra time-confidence parameter that can both give confident time intervals and more effective rules. This time-confident rule set can be used with event monitoring engines such as CEP systems. A simulation study was performed to test of our method with real-world alarm data from a large mobile telecom operator. Our method improves the accuracy of alarm rule finding space and can significantly reduce the number of registered alarm rules (e.g. 9/25 or 64%).

References

1. M. Liu, E. Rundensteiner, K. Greenfield, C. Gupta, S. Wang, I. Ari, and A. Mehta, "E-cube: multi-dimensional event sequence analysis using hierarchical pattern

query sharing," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD '11, (New York, NY, USA), pp. 889–900, ACM, 2011.
2. S. Wallin, "Chasing a definition of alarm," *Journal of Network and Systems Management*, vol. 17, pp. 457–481, 2009.
3. T. Li and X. Li, "Novel alarm correlation analysis system based on association rules mining in telecommunication networks," *Inf. Sci.*, vol. 180, pp. 2960–2978, Aug. 2010.
4. P. Wu, R. Bhatnagar, L. Epshtein, M. Bhandaru, and Z. Shi, "Alarm correlation engine (ACE)," in *Network Operations and Management Symposium, 1998. NOMS 98., IEEE*, vol. 3, pp. 733–742, IEEE, 1998.
5. S. Laxman and P. S. Sastry, "A survey of temporal data mining," in *SADHANA, Academy Proceedings in Engineering Sciences*, 2006.
6. R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, (Washington, DC, USA), pp. 3–14, IEEE Computer Society, 1995.
7. H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, pp. 259–289, Jan. 1997.
8. R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993* (P. Buneman and S. Jajodia, eds.), pp. 207–216, ACM Press, 1993.
9. P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, "Rulegrowth: mining sequential rules common to several sequences by pattern-growth," in *26th ACM Symposium On Applied Computing, SAC'11*, pp. 956–961, 2011.
10. G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series," in *KDD'98*, pp. 16–22, 1998.
11. S. Lallich, O. Teytaud, and E. Prudhomme, "Association rule interestingness: Measure and statistical validation," in *Quality Measures in Data Mining*, vol. 43 of *Studies in Computational Intelligence*, pp. 251–275, Springer Berlin Heidelberg, 2007.
12. P.-H. Wu, W.-C. Peng, and M.-S. Chen, "Mining sequential alarm patterns in a telecommunication database," in *Proceedings of the VLDB 2001 International Workshop on Databases in Telecommunications II*, DBTel '01, (London, UK, UK), pp. 37–51, Springer-Verlag, 2001.
13. P. F. Viger, U. Faghihi, R. Nkambou, and E. M. Nguifo, "CMRules: An efficient algorithm for mining sequential rules common to several sequences," *Elsevier, Knowledge Based Systems*, vol. 25, pp. 63–76, Jan. 2012.
14. "EsperTech inc.." `http://www.espertech.com`, 2014. [Online; accessed March-2014].
15. J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods," *Journal of Intelligent Information System*, vol. 28, no. 2, pp. 133–160, 2007.
16. M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Machine Learning*, vol. 42, pp. 31–60, Jan. 2001.