REAL-TIME EVENT CORRELATION AND ALARM RULE MINING MODELS FOR COMPLEX EVENT PROCESSING SYSTEMS

A Thesis

by

Ömer Faruk Çelebi

Submitted to the
Graduate School of Sciences and Engineering
In Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in the
Department of Computer Science

Özyeğin University
August 2013

REAL-TIME EVENT CORRELATION AND ALARM RULE MINING MODELS FOR COMPLEX EVENT PROCESSING SYSTEMS

Approved by:

Professor İsmail Arı, Advisor
Department of Computer Science
Özyeğin University

Professor Hasan Sözer
Department of Computer Science
Özyeğin University

Professor Ekrem Duman
Department of Industrial Engineering
Özyeğin University

Date Approved: 21 August 2013

To my lovely son, Hezarfen Yahya Çelebi

ABSTRACT

World is creating the same quantity of data every two days, as it created from up until 2003. Evolving data streams are key factor for the growth of data created over the last few years. Streaming data analysis in real-time is becoming the fastest and most effective way to get useful information from what is happening right now, thus allowing organizations to take action quickly when problems occur or to detect new trends to improve their performance. Data stream analytics is needed to manage the data currently produced from applications such as sensor networks, measurements in network monitoring, mobile traffic management, web click streams, mobile call detail records, social media posts/blogs and many others. Stream data analytics is hard because data are temporally ordered, fast changing, massive and potentially infinite. In order to cope with the challenges of data stream mining, in this thesis two main contributions are discussed. Both of them summarize the high volume streaming data and present meaningful, actionable information to end users. The first one is finding "event correlations" over the data stream pairs on real GPS data of public transportation buses. The second one is alarm sequence rule mining, with a new parameter called "time confidence", that helps automatically set time-window values for registered rules and also reduces the generated alarm rule count.

ÖZETÇE

Dünya, her iki günde bir 2003 yılına kadar ürettiği veri miktarı kadar veri oluşturmaktadır. Gelişen veri akışları son birkaç sene içerisinde üretilen verinin büyümesindeki en önemli etkendir. Gerçek zamanlı olarak yapılan veri akışı analizi, şu an gerçekleşenler hakkında yararlı bilgi edinilmesini sağlayan en hızlı ve en etkili yol olması, organizasyonların ortaya çıkan problemler için hızlıca aksiyon almalarına ya da yeni trendleri keşferek kendi performanslarını arttırmalarına yardımcı olmaktadır. Gerçek zamanlı veri akışı analizi, sensör ağları, ağ izlenmesindeki ölçümler, mobil trafik yönetimi, web gezintisindeki tıklama akışları, mobil arama detay kayıtları, sosyal medya iletileri/günlükleri ve benzeri daha birçok uygulamalardan üretilen verinin yönetimini yapmak için gereklidir. Veri akışı analizi zordur çünkü veri akışları geçici olarak sıralı, hızla değişen, yığın ve potansiyel olarak sonsuzdurlar. Veri akışı madenciliğindeki bu zorluklarla başa çıkabilmek için bu tezde iki çalışma yapılmıştır. Her iki çalışma da yüksek miktardaki veri akışını, son kullanıcılar için anlamlı ve aksiyon alınabilir şekilde sunmaktadır. Birinci çalışmada, toplu taşımada kullanılan otobüslerin gerçek GPS veri akışı çiftleri üzerinde "olay ilişkilerinin" bulumasıdır. Diğeri ise "zaman güvenilirliği" olarak adlandırılan yeni alarm ardışıl kural madenciliği parametresidir. Bu parametre kayıt edilen kurallar için pencere zamanı sağlar ve aynı zamanda üretilmiş kuralların doğru bir şekilde azaltılması üzerinde etkisi vardır.

ACKNOWLEDGEMENTS

Time is passing quickly and it has been more than three years since I have started my master work. This thesis could not have been completed without the support, encouragement and guidance of my family, friends and advisor.

First of all, I would like to thank my advisor Dr.İsmail Arı for serving as a mentor during my graduate studies.I was extremely fortunate to study with him. He opened new doors for me with new research topics which have become the fundamental parts of this dissertation. He always made time for me, had long discussions with me and gave me invaluable advice. His help during my studies kept me motivated and added new and better dimensions to my research. I am grateful to him. He also gave me a large amount of freedom during my graduate study.

I am also grateful to my thesis committee members Prof.Hasan Sözer and Prof.Ekrem Duman for their valuable comments and time.

Finally, my wife, Kübra Çelebi, has always given me strength throughout my entire life especially during my studies. Her love and support have guided me through the years. Dad and Mom, Mehmet and Şimşat Çelebi and parents-in-law, Muzaffer and Canan Gökçe you have always supported me and been there for me. I owe you a lot for all the things I have achieved in life. I would also like to thank my brothers and sisters for their continuous support and encouragements. This thesis is a tribute to you all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivations for Real-Time Data Stream Analysis

We are seeing increase in data stream applications over the last years. The world is creating the same quantity of data every two days, as it created from up until 2003. The applications that generate data include computer network monitoring, Radio Frequency Identification (RFID)-based supply chain and traffic management systems, route tracking, alarm management, e-trading, online financial transactions, web click-streams, mobile communication applications and civilian or military sensor networks. All of these applications are mission-critical for related organizations and require real-time stream processing to detect simple or complex events, so that strategic decisions can be made quickly. Signals and patterns inside data streams allow organizations to take quick action before/when problems occur or to detect emerging trends to improve system performance. Currently, many organizations still use Database Management Systems (DBMS) in an ad-hoc fashion for data stream analytics. They face performance or technical problems with continuous and time-window based analysis over high-volume data streams, since DBMS architecture was designed for first storing the data and then analyzing it. The requirements are not met and the expensive investments become inadequate. An emerging system architecture called Data Stream Management System (DSMS) is better suited to address the analysis needs of emerging data stream applications.

Today's database architectures have a 50 year old history. Several significant milestones such as definition of the relational model and structured query language (SQL), implementation of the first engines, query plan optimization, and distributed

design concepts have been completed and are still being advanced. These milestones in database history were followed by integration of data warehouses, triggers, data mining algorithms, and addition of numerous middleware and software components into the database suites. However, these gigantic and complex architectures were optimized for analysis of relatively static data (and even only for hard-disk drives), therefore they could not provide an effective solution for the real-time analysis needs of high-volume data stream applications whose number has been increasing due to the developments in the Internet, sensor and mobile technologies. For these reasons several DSMS systems have emerged [3] [4] in the last ten years and they have been following a similar advancement path with the traditional DBMS. A comparision of DBMS versus DSMS is given in Table 1.

Table 1: Principles of DBMS and DSMS(Adopted from [1])

DBMS	*DSMS*
Persistent data (relations)	Volatile data streams
Random Access	Sequential access
One-time ad-hoc queries	Continuous queries
Unlimited secondary storage	Limited main memory
Relatively low update rate	Extremely high update rate
Assumes exact data	Assumes outdated or inaccurate data
Plannable query processing	Variable data arrival and characteristics

After demonstrating simple operations such as filtering and aggregations over streams some of these leading DSMS were incorporated into real industrial applications. Parallel to the developments in DSMS, software programs called Complex Event Processing (CEP) engines, which internally utilize a rule engine or a Finite State Automaton (FSA), have emerged within the last 10 years.

These concepts and data stream processing also attract mobile telecommunication operators. These operators have lots of application areas for deploying DSMS or CEP engines. They have large networks which consist of thousands of devices/network

equipments that generate different sets of alarms.

The goal in this thesis is to design and implement a real-time CEP engine that can be used for alarm management especially in wireless cellular networks for streaming data. The innovative aspect of the design is the integration of stream correlation and stream event sequence mining capabilities (that will be described in detail) into the same advanced CEP engine. Successful implementation of the proposed system will satisfy the real-time or near real-time data analysis requirements of the mobile cellular data stream applications.

1.2 Wireless Sensor and Cellular Networks

Wireless Sensor Networks (WSN) are used for real-time monitoring of physical environments. They help higher-level applications collect relevant data that can be transformed into actionable information. These applications include earthquake monitoring, asset tracking, traffic management, national security, green data centers [5], and recently regulatory hygiene-compliance tracking in hospitals [6]. People managing or using these applications are interested in detecting and even predicting concise "special events" (e.g. anomalies) upon which they can take an application-specific action.

Events are semantically different from primitive numeric sensor readings. For example, an event can refer to a numeric threshold violation or a more complex pattern such as an ordered (possibly nested) sequence of any datum. Finding complex event patterns in high-speed, unbounded, bursty data streams can be as challenging as finding a needle in a haystack. Sometimes checking only the "existence" of a simple reading in the stream may be of interest and sometimes we look for the "absence" of an event instead of its existence. Doing these becomes hard when the streams come from distributed sources. Ability to aggregate, order, join or correlate streaming data is the key to detecting many of these complex situations. Event correlation engines,

some of which will be described here, help us describe these scenarios and find event patterns effectively. However, even the state-of-the-art systems including Hadoop [7] cannot cope with today's real-time and distributed event processing challenges.

Wireless cellular networks also consist of thousands of networks equipments. 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 their time for analyzing and interpreting the daily alarm information. In addition, faults that can interfere with routine services offered by the network operator decreases 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 and 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.

Most of these stream processing and analytics challenges can be addressed through the use of Data Stream Management Systems (DSMS) [3] [4] in the data pipelines of organizations. Therefore, enterprises increasingly utilize these systems and extend their basic filtering facilities with complex online analytics and mining capabilities. The resulting software tools are sometimes called Complex Event Processing (CEP) engines in the literature [2]. The benefits of using DSMS and CEP systems are at least three-fold:

1. They can eliminate unwanted data early in the pipeline, saving further CPU, memory, storage and energy costs.

2. They can turn raw data into actionable information quickly, thus helping businesses catch profitable opportunities or avoid losses due to fraud or operational

inefficiencies.

3. They can catch transient or emerging patterns, which never show up in an offline data mining analysis.

1.3 Thesis Problem Statement and Contributions

The contributions of this thesis are briefly as follows: we implement and demonstrate that both statistical analysis (e.g. stream correlation) and data mining (e.g. rule mining) can be implemented over the same system and be used for different real-time applications. We also show that for both analytic and mining tools, the semantics of time-windows (type and size) can have a great impact on both the performance and usability in different applications.

The rest of the thesis is as follows. In Chapter 2, stream correlation for sensor data is discussed. Basic filtering and aggregation over streams, window types and correlation are also explained in detail. In Chapter 3, we discuss the background and terminology for alarm domain as well as sequential alarm correlation and the proposed alarm mining method. Chapter 4 concludes the thesis and discusses some of the future work.

CHAPTER II

A STREAM CORRELATION MODEL FOR SENSOR DATA

2.1 Challenges of Sensor Data Processing

Consider the data sample in Table 2 pertaining to only one vehicle collected from a real geo-tracking system. For this vehicle with the unique id (00-123) the data shows the geolocation (longitude and latitude), speed and time information for every 20 seconds. However, due to intermittent disconnects or noise in the channel many data fields are prone to different types of errors. For example, while a normal value for the longitude and latitude fields would have 8 digits (e.g. 28.866.064, 41.052.856) we see that many entries lost several of their least-significant digits. Similarly, we find that the highly-varying speed information may also be erroneous. Note that some rows can be completely missing. For example, at minute 12/8/2009 7:26 only one measurement was recorded instead of three. Other potential data anomalies include accuracy errors and out-of-order arrivals [8]. This is only one data stream and yet there are thousands of vehicles and millions of objects that need to be tracked in WSNs. Mass transportation administrators want the flexibility to be able to accurately track a single vehicle or average recordings from all vehicles on a certain route or certain region.

Streaming data from sensors is also prone to errors, which reduces its "veracity" (or accuracy). The tuples can be missing, broken, out-of-order, or they can have wrong values. Table 2 shows sample data from a real bus tracking application, which shows multiple such cases. The system accidentally dropped zeros from the least significant digits of longitude and latitude of bus locations, missed some tuples and

possibly inserted inaccurate speed values (0, 1 km/h).

Table 2: Sample Data From a Real Geo-Tracking Application

ID	$LONGITUDE$	$LATITUDE$	$SPEED$	$DATEandTIME$
00-123	28,863,169	4,105,348	42	12/8/2009 7:23
00-123	2,886,469	41,052,845	3	12/8/2009 7:23
00-123	28,866,064	41,052,856	26	12/8/2009 7:23
00-123	28,867,975	410,522	37	12/8/2009 7:24
00-123	2,886,879	4,105,189	1	12/8/2009 7:24
00-123	28,869,068	41,051,792	6	12/8/2009 7:24
00-123	28,869,884	41,051,376	16	12/8/2009 7:25
00-123	28,870,121	41,051,258	0	12/8/2009 7:25
00-123	2,887,055	41,051,044	16	12/8/2009 7:25
00-123	28,870,613	4,105,191	15	12/8/2009 7:26
00-123	28,868,597	4,105,249	46	12/8/2009 7:27
00-123	28,866,816	4,105,319	19	12/8/2009 7:27
00-123	288,657	41,053,898	20	12/8/2009 7:27

Overall, the challenges and issues [4] [9] in managing WSNs data streams include:

1. Limitations on communication range, power, CPU and memory of the wireless sensors and sensor networks resulting in broken data and out-of-order arrivals

2. Need for real-time data cleansing and sanity checking and associated challenges

3. Need to eliminate duplicate or unnecessary data to save resources without destroying the essence of information carried inside the streams and without missing critical events

4. Finding correlations over high-speed raw or aggregated or sketched/summarized data streams [10] [11] [12]

5. Having large number of streams to join and correlate; Holistic querying and monitoring over distributed streams

6. Differences in observed data types (integer, float, string, date-time) and varying data values (0-1, 28868597, "BUS- 00-130", etc.)

7. Widely-varying data sampling frequencies (from microsecond, to seconds-minutes, to hours-days-months) among different sensor types.

8. Lack of trust among organizations for sharing raw data. The need to operate over encrypted or compressed data (e.g., using fully homomorphic functions).

9. Effective data visualization.

2.2 Basic Filtering and Aggregation Over Streams

DSMS engines provide effective queuing, scheduling, time and count-window support, and fast in-memory processing of high-speed, continuous, unbounded data streams [3]. They parse, optimize and execute queries written in declarative languages such as Event Processing Language (EPL) in Esper [2]. EPL syntax and semantics are quite similar to that of Structured Query Language (SQL) in databases, but there are additional clauses such as WINDOWs to support sliding or tumbling window-based analytics over data streams. Figure 1 shows these two types of windows.

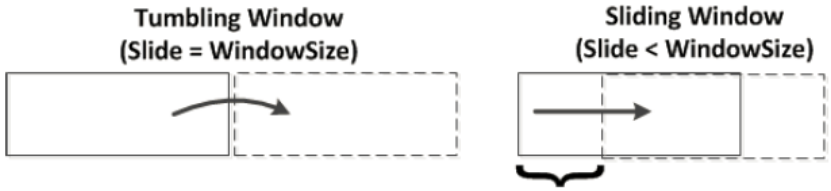


Figure 1: Tumbling vs. Sliding Window semantics

Sliding-time windows are used to buffer event tuples whose occurrence times fall within a certain time period (e.g., last 1 minute) and to replace events that are older than the time window. The window will move or slide in time with a EPL queries can be used for continuous filtering (e.g., **SELECT** x,y **FROM** Stream $< x, y, z >$ **WHERE**) as well as aggregations: algebraic (COUNT, SUM, AVERAGE)

or holistic (MIN, MAX). Complex aggregation functions such as TOP-K, DISTINCT, QUANTILES, and SKYLINE can also be found or implemented.

Sliding windows enable users to limit the number of events considered by a query. As a practical example, consider the need to determine all accounts where the average withdrawal amount per account for the last 4 seconds of withdrawals is greater then 1000. The statement to solve this problem is shown below.

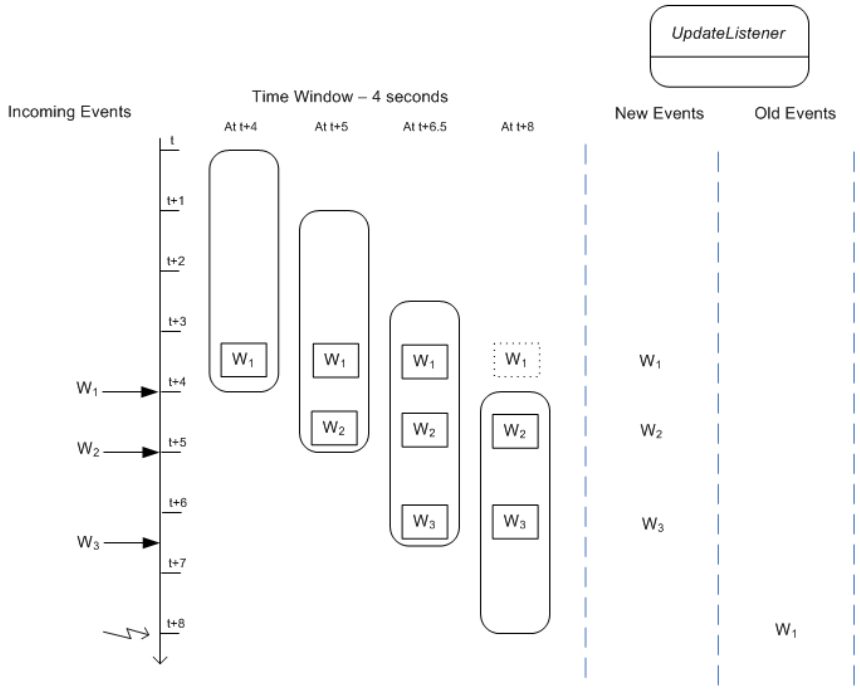
```
SELECT account, avg(amount)
FROM Withdrawal.win:time(4 sec)
GROUP BY account
HAVING amount >1000
```

The Figure 2 serves to illustrate two types of the functioning of a time window. For this figure, it is assumed that a query simply selects the event itself and does not group or filter events.

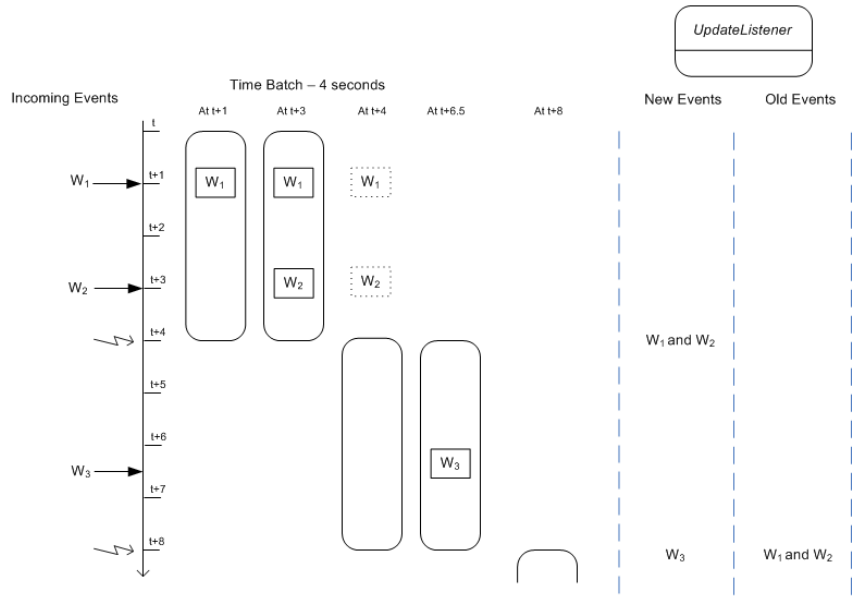
```
SELECT *
FROM Withdrawal.win:time(4 sec)
```

The figure starts at a given time t and displays the contents of the time window at t + 4 and t + 5 seconds and so on.

The activity as illustrated by the Figure 2(a) can be summarized as : At time t + 4 seconds an event W1 arrives and enters the time window. At time t + 5 seconds an event W2 arrives and enters the time window. The engine reports the new event to update listeners. At time t + 6.5 seconds an event W3 arrives and enters the time window. The engine reports the new event to update listeners. At time t + 8 seconds



(a)



(b)

Figure 2: Output example for a statement with (a) Sliding Window (b) Tumbling Window (Adopted from [2])

event W1 leaves the time window. The engine reports the event as an old event to update listeners. On the other hand, the tumbling window buffers events and releases them every specified time interval in one update. Time windows control the evaluation of events, as does the length batch window. The Figure 2(b) illustrates the functioning of a tumbling window. For this Figure 2(b), it is assumed a simple query as below:

```
SELECT *
FROM Withdrawal.win:time_batch(4 sec)
```

The Figure 2(b) starts at a given time t and displays the contents of the time window at t + 4 and t + 5 seconds and so on. The activity as illustrated by the diagram can be summarized as: At time t + 1 seconds an event W1 arrives and enters the batch. No call to inform update listeners occurs. At time t + 3 seconds an event W2 arrives and enters the batch. No call to inform update listeners occurs. At time t + 4 seconds the engine processes the batched events and a starts a new batch. The engine reports events W1 and W2 to update listeners. At time t + 6.5 seconds an event W3 arrives and enters the batch. No call to inform update listeners occurs. At time t + 8 seconds the engine processes the batched events and a starts a new batch. The engine reports the event W3 as new data to update listeners. The engine reports the events W1 and W2 as old data (prior batch) to update listeners.

CEP is an important modern application framework to submit complex queries to track events sequences that can provide a matching for a given pattern. Moreover, sequentiality is the primary way to relate events to each other in CEP systems. In this thesis, we use a CEP engine called Esper, which has been developed by EsperTech as an open source product [2]. The Esper CEP engine uses EPL (event processing language) which is a SQL-like language. Event stream queries in Esper follow EPL syntax that provide the aggregation, joining, windows and analysis functions for use

with events streams. In SQL-based continuous query language, the output relation in time-based sliding windows on a ordered stream S is defined as relation over time by sliding an interval of size W time units in order to capture the last W time units of the ordered stream. On the other hand, in tuple-based window the output relation is defined over time by sliding a window of the last N tuples of an ordered stream [13].

The input to a CEP system is event streams generated by external processes. In order to detect sequences of correlated events, i.e. event patterns, users have to register or subscribe to running queries. CEP queries have the following format [14]:

PATTERN composite event expressions
WHERE value constraints
WITHIN time constraints
RETURN output expression

where *composite event expressions* are set of rules to match an event pattern in the data stream, *value constraints* are the predicates on the composite events, *time constraints* describe predefined time interval where matching event patterns must occur and *output expression* defines the output stream from the pattern query.

2.3 Correlation over Streams

In this section, we describe Pearson Product Moment Correlation (PPMC) [15] operator over streams and show its application to route-matching over GPS data streams. Briefly, correlation is the covariance of two variables divided by their standard deviations. The correlation value can change between [-1,+1], where +1 denotes a high-positive correlation, 0 denotes no correlation and -1 denotes high-negative correlation. For this thesis, to correlate a bus to its previously recorded route or to ensure that two vehicles move together,continuous queries on vehicle latitudes and

longitudes is implemented and used the following (Figure 3 shows longitude):

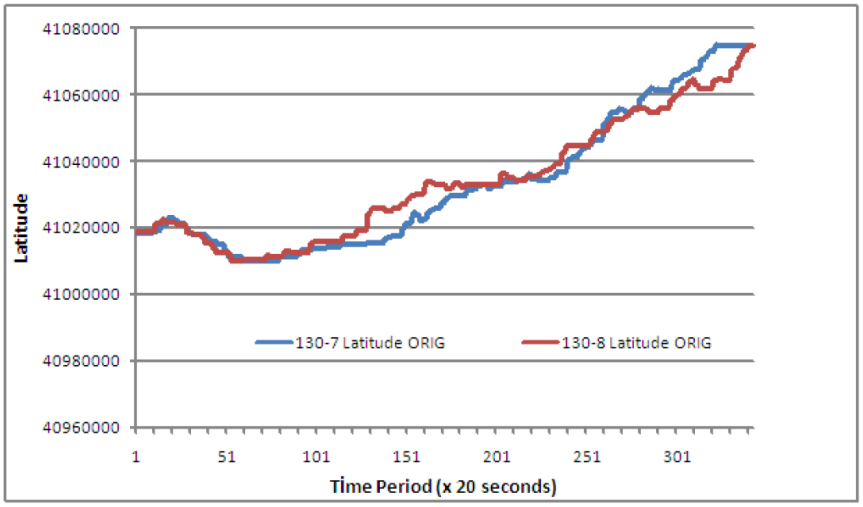
```
SELECT CorrelationLatitude
FROM VehiclePairStream
WIN   length(50).stat:correl(a.long, b.long)
```

```
SELECT CorrelationLongitude
FROM VehiclePairStream
WIN   length(50).stat:correl(a.lat, b.lat)
```

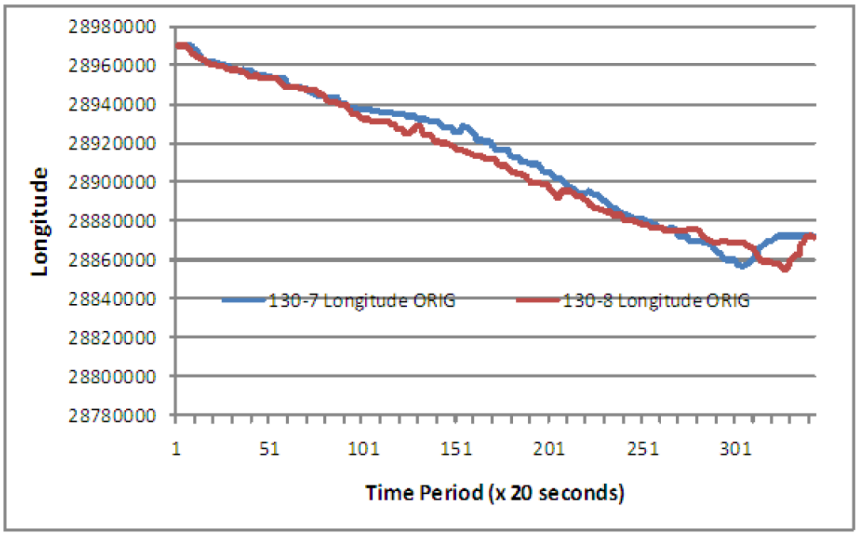
where the VehiclePairStream is constructed by joining the two streams (a, b). Figure 3 shows the time-series data for longitudes collected for two different busses on the same route. The two stream variables could also belong to the current bus under investigation and its pre-recorded route obtained by averaging the longitudes of the busses that travel daily on the same route. The goal is to continuously track buses and detect anomalies in real-time such as group separation, out-of-route movements or extreme traffic delays.

2.4 Bus data evaluation

GPS data from a single bus (00-130) moving along the same route on two different days (denoted as 7 and 8) was used for the evaluations in this section. Two different buses that move along the same route on the same day or different dates could also be used. We expect to find high correlations for vehicles that go over the same route, share sub-routes, or move together. Because the correlation is tracked over streaming data we will be able to detect emerging patterns such as traffic congestions or unique violations in real-time and publish short alerts as interesting events to watch to the related people. Note that some of this data may have to be stored or buffered before



(a)



(b)

Figure 3: Moving vehicle coordinates over the same route on different days and start times (a) Latitude (b) Latitude .

(a)

(b)

Figure 4: Tracking vehicles over Google Maps and correlating sensor readings to automatically detect which vehicles and routes are related and how (a) The complete route for the same bus on two consequtive days is tagged and correlated (b) Zooming into one of the subjections of the route. Satellite data is dispersed (accuracy is lost) when vehicles are around buildings

being correlated with others as well. The experiments were executed by running the correlation queries over the open-source Esper engine on a personal computer with Intel i5 processor and 3GB memory.

For visual confirmation we marked the movements of vehicles on Google Maps as shown in Figure 4. Each mark in this figure shows where each vehicle was during a specific recording. This also allows us to compare how a single bus moved at different times along the same route or two different buses moved with respect to each other - faster or slower- on the same path. Figure 3 shows the extracted longitude and latitude information from the two routes. These are the two values that are being separately

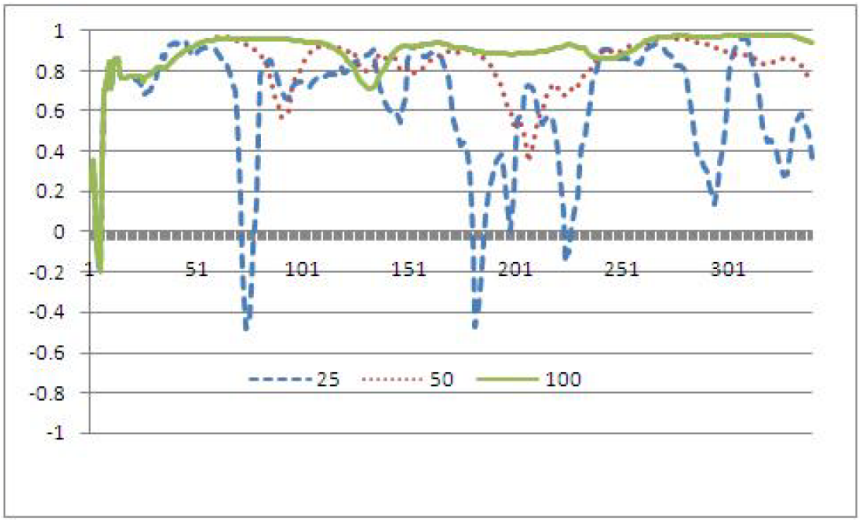


Figure 5: The effect of changing sliding window size on the correlation coefficient

correlated, respectively. Correlation coefficient is our statistical tool for finding event correlations (e.g., group asset tracking or route enforcement). To test its sensitivity, we investigate how changing the sliding window size affects the coefficient value and the processing performance (average latency of getting the results). Ultimately, we dont want to depend on any magic parameters.

Figure 5 shows the effect of window size on the correlation coefficient. Smaller window sizes have less data to compare, therefore when buses move differently with respect to each other the correlation value can drop sharply for that period. For larger window sizes like 50-100 this effect is compensated for, as one bus usually catches up with the other (or the same bus compensates for its transient delay over the same route at different times). To reduce the amount of output produced, we could use tumbling windows which only publish results at the end of a time or count period. A tumbling window is basically a discrete version of the continuous sliding window.

Figure 6 shows the correlation results for the tumbling windows. The results are similar to their corresponding sliding windows (on average higher for the larger windows), but they are published less frequently. While tumbling compensates for some of the jitter, the cost to calculate results increases as the window size increases

	Window Size		
Time	25	50	100
25	0.7725		
50	0.9293	0.9101	
75	0.1714		
100	0.7335	0.7469	0.9528
125	0.8016		
150	0.5505	0.7989	
175	0.5594		
200	0.1531	0.6363	0.8842
225	0.2692		
250	0.8886	0.8978	
275	0.8927		
300	0.3465	0.9086	0.9726
325	0.4321		
Average	0.577	0.8164	0.9365

Figure 6: Calculating correlations using tumbling windows

as shown in Figure 7. Our future work includes running these queries over our high-end IBM Blade HS22 servers and testing performance over multi-core and distributed resources.

Figure 7 shows that changing the sliding window size does not affect the output tuple (correlation result) latency. This is because the each component of the correlation operation can be done incrementally. For example, we can maintain a running average of the scalar values by using the formulas:

$$\text{TotalValNew} = \text{TotalValOld} \quad \text{ValueOut} + \text{ValueIn}$$

$$\text{NewAve} \quad = \text{TotalValNew} \; / \; \text{WindowSize}$$

For tumbling windows the delay increases logarithmically as the window size also increases logarithmically (shows linear on log-log scale), because the data is collected and processed at once for the time interval at the end of that time period.

Figure 8 illustrates different uses of lat-long correlation information on a city map.

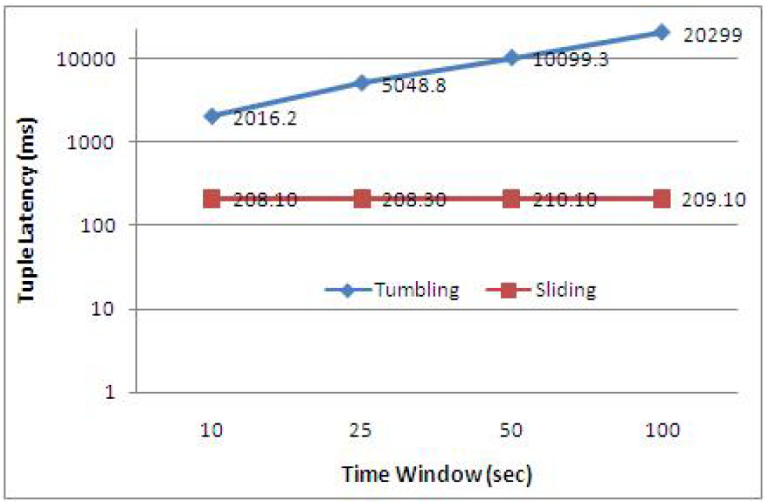


Figure 7: Performance comparison of sliding and tumbling windows at different window sizes.

Using statistical correlation, a moving vehicle can be correlated to its assigned route to assert in-route movement (a correct reference path is recorded beforehand), multiple vehicles on the same route can be correlated to spot erratic driving behavior, or assets moving in a supply-chain can be grouped together to assure intact delivery of goods to the distribution centers or retailers. However, we should carefully understand the issues with the new operator (i.e. Correlation) before we can use it for complex event detection. For example in Figure 8, two vehicles that are moving in different parts of the city (shown in rectangular boxes) on a very similar trajectory can have high correlation values since the latitude and longitude vectors are the same except only a spatial shift. Additional domain specific information may have to be used to detect whether these two vehicles are actually the same vehicle, are on the same route or belong to an asset group. Applying range queries (like the boxes in Figure 8) may be useful for assuring spatial relations.

The other challenge is related to the temporal component of the correlation. Vehicles on the same route pass from same points at different times and possibly move along the same trajectory with varying time-scales. Time-shifting and time scaling is necessary to associate vehicles with their routes and other vehicles on the same

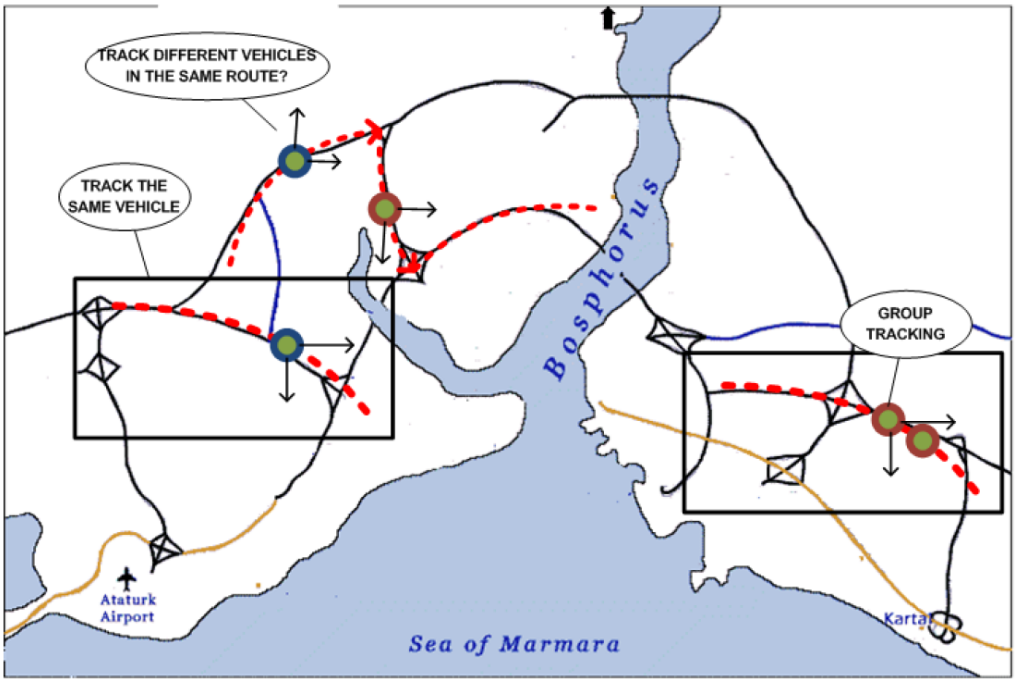


Figure 8: Detecting and enforcing in-route or group movement of vehicles using statistical correlations

route. We applied a time shift manually for the data used in this thesis, but we plan to use either the Regressions or B-Splines techniques to create feature vectors for the routes and automatically correlate time shifted and time-scaled events series in real-time. Finally, running these correlations over broken data is a big challenge and the only-solution is to do real-time data cleaning, preprocessing over the stream.

To summarize, correlation over geo-spatial streaming data is quite insensitive to spatial shifts, but is very sensitive to time shifts and broken data. We observe that running separate regressions on each variable can both correct broken fields and help us estimate missing or wrong values.

2.4.1 Results and Discussion

Figure 5 shows the results of correlation for different sliding and tumbling window sizes $(25 - 50 - 100)$. If the correlation C is lower than a certain threshold (e.g., C ¡ 0.8) an alarm can be generated. The bus has either gone out of route or is not moving

timely, both of which denote anomalies. Note that smaller window sizes have less data to compare, therefore when buses move even slightly different with respect to each other, the correlation value drops sharply for that period. Therefore, small windows result in high false positive rates. For larger window sizes like 50-100 this effect is compensated for, as one bus usually catches up with the other (or the same bus compensates for its transient delay over the same route at different times). To reduce the amount of processing and output produced, we could use tumbling windows which only publish results at the end of a time or count period. We found that tumbling window is basically a discrete version of the continuous sliding window and similar correlation results are published by both. Therefore, we skip tumbling window results for brevity. As shown in Figure 7, changing the sliding window size does not affect the processing latency, since parts of the correlation formula are calculated incrementally. For tumbling windows the delay increases with window size, because all the data that is collected until the end of a time interval is processed at once. This finding is in line with the motivations for fast incremental updates (FUP) used in finding frequent itemsets over streams [16] [17] [18].

CHAPTER III

AN ALARM RULE MINING MODEL FOR CELLULAR DATA

3.1 Background and Related Work for Alarm domain

In the analysis of alarm flows in telecommunication networks, there are different approaches for dealing with the alarm correlation problem [19] [20]. Some of the approaches focus more on implementing the alarm correlation engine [21], while others focus on alarm modeling and validation scenarios [20].

There is also numerous prior work on temporal data mining that observe frequent patterns in a sequence database (see [22] for a survey). Some of these papers have considered only Top-K event sequence detection in an alarm database [23] [24]. Algorithms such as GSP [23] and WINEPI [24] were the first to apply Apriori algorithm [25] to find sequential association rules temporally. These algorithms require a user-defined sliding-time window duration to traverse the data.

Most of the sequential data mining methods proposed so far are based on discovering order relations between events [23]. These methods are based on finding the frequencies of event sequences and generating the rule candidates against the database [26]. Some have concentrated on prediction problems for sequential rule mining on several different application domains [26] [27]. However, these algorithms do not focus on time-interval differences between events (item sets) in a rule. 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 [28]. However, this paper does not discover the potential time interval patterns between events for setting the urgent window size.

CEP is an important approach for applications that need real-time responses for event streams. CEP systems have become very popular for searching for sequences of incoming events for occurrences of user-specified event patterns or pattern matchings. CEP can also provide a strong query language, powerful mechanisms to create complex events from elementary ones and potential performance improvements (see [29] for a detailed survey on CEP).

Alarm standards like 3GPP [30] and X.733 [31] defined by ITU-T, define the alarm protocols and parameters. X.733 is the standard for alarm interfaces where almost all alarms adhere to the definitions that are defined in [31]. According to X.733 protocol, an alarm typically contains the attributes such as equipment name, device type, alarm time, alarm level, alarm type, interface type, alarm severity, alarm name identifier, equipment address, etc. In this thesis, we extract equipment type (e.g., network elements such as MGW, MSC, NE3G, etc), device type (e.g., devices in each network elements), alarm name identifier (i.e. alarm message) and alarm time to form an alarm event. An example of alarm generated by some of the network components are shown in Table 3. An alarm correlation framework should adapt to changes in alarm messages and also to component changes when extensions to the network structure are performed.

3.2 Sequential Alarm Correlation for CEP

Real-time performance monitoring and optimization of the measured alarms require a flexible alarm monitoring framework taking into account numerous alarm types, alarm arrival frequencies and time intervals. In order to create an effective alarm performance monitoring and optimization functions that rely on large-amounts of event-based triggering data, a framework architecture is needed. This framework should handle the received alarm data efficiently, process and forward this information before the main event occurs.

In this section, we define a method and propose a new *time-confidence* metric for sequential rule mining algorithms in the domain of alarm correlation. Our proposed method differs from previous works by relying on both input of time confidence values by the user as well as determining the best sliding time window interval for the CEP system. The main advantage of this method is to enable alarm handling with minimal operating costs and human intervention. Therefore, this method achieves a more intelligent and automated solution to correlate alarm prediction events.

The motivation for this work was the lack of an accurate and reliable alarm rule discovery and reduction framework. Real time alarm tracking or event sequence detection systems [6] 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 work, we try to answer the following questions:

1. How can the discovered alarm sequences be reduced effectively?

2. How can the constructed alarm rules be registered into a real-time warning system more efficiently?

In order to answer 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.

3.3 Proposed Sequential Alarm Correlation Mining Method

The problem which is addressed by the proposed method is to find the complete set of patterns that satisfy a given minimum time-confidence threshold in the alarm transaction database. The time-confidence of a pattern is the maximum time interval for

registration of query time into the CEP system. It is selected as the maximum occurrence of a specific time interval containing the discovered pattern from the generated alarm rule database.

With this method, the minimum time-confidence threshold is used to prune the infrequent time intervals between the series of events of a generated sequential pattern of a sequential rule mining algorithm. Based on this threshold, if a pattern has a large time interval variance, the candidates for the registration into the CEP system will be small or even this rule will not be registered into the CEP system if the minimum time-confidence threshold is high.

In this thesis, we consider the problem of sequential rule mining for correlated sequences as follows: Let a *sequence database* $SeqT$ be the set of ordered event sequences of denoted by $SeqT = \{c_1, c_2, ..., c_k\}$ 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 an itemset $I = \{i_1, i_2, ..., i_n\}$ where $X_1, X_2, ..., X_m \subseteq I$ [26]. The length of the sequence set $SeqT$ is denoted by $|SeqT|$. Every item i_k in an itemset X_i has a special attribute called *timestamp*, denoted as $i_k.time$, which records the time when the item occurred. A sequential rule $r \in R$ such as $(X_i \Rightarrow X_j)$ is defined as a relationship between two itemsets $X_i, X_j \subseteq I$ such that $X_i \cap X_j = \emptyset$ and X_i and X_j are non-empty, where R is the set of rules and items in X_i occur before the items in X_j. The ordering of items within X_i and X_j are not restricted.

In sequential rule mining [24] [25] [27], two different measures can be defined : the *sequential support* (denoted here as *support* in short), given as $sup(X_i \Rightarrow X_j) = sup(X_i \blacksquare X_j)/|SeqT|$, and *sequential confidence* (denoted here as *confidence*) given as $conf(X_i \Rightarrow X_j) = sup(X_i \blacksquare X_j)/sup(X_i)$ [26] for ordered events. Here $sup(X_i \blacksquare X_j)$ denotes the number of occurrences of $X_i \blacksquare X_j \in SeqT$ where all items of X_i appear before all the items of X_j.

In addition to the above, this thesis introduces a new metric called *time confidence*

which is defined as:

$$time\_conf(X_i \overset{T(r)}{\Longrightarrow} X_j) = \frac{sup(X_i, X_j, T(r))}{sup(X_i \blacksquare X_j)} \tag{1}$$

where $T(r)$ is the time interval set between two successive itemsets X_i and X_j, given as $T(r) = \{T_1, T_2, ..., T_l\} = \{T_i | T_i \in |X_{j_{[i_n.time]}} - X_{i_{[i_1.time]}}|\}\}$, where l is the distinct number of time interval differences for the rule r and $sup(X_i, X_j, T(r)) = |\{c_i | c_i \in SeqT \bigcap (X_i \overset{T(r)}{\Longrightarrow} X_j)\}|$ represents the number of transactions were each transaction contains rule r for each time interval of $T(r)$.

Our goal 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 as 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 using only min support and confidence values. These significant rules with time confidence can now be placed into the Complex Event Processing (CEP) engines for accurate and reliable real-time alarm management.

The proposed method extracts more useful information than traditional sequential data mining techniques by considering an additional *time confidence* value for each associated rule. These important rule decisions can also be an input to Data-Stream Management System (DSMS) based systems such as CEP engines.

The procedure for the proposed method is explained in Algorithm 1. The goal is to select rules that have support, confidence as well as time-confidence values greater than user-defined min_supp, min_conf and min_time_conf thresholds, respectively. A rule that has support greater than a min_supp is considered as *frequent*, a rule with a confidence greater than min_conf is *confident* and a rule with a time-confidence higher than min_time_conf is considered to be *time_confident* or significant.

Note that at step 4 of Algorithm 1, T_{time_conf} value is obtained for each rule. This

Algorithm 1 Proposed method

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

1. Scan the database $SeqT$ to discover the sequential rules $r \in R$ using a sequential rule mining algorithm (e.g., RuleGrowth [26]), GSP [23], PREFIXSPAN [32], SPADE [33] based on min_sup and min_conf.

2. Calculate the event time interval differences of all sequential alarm rules and record them in $T(r) = \{T_1, T_2, ..., T_l\}, \ \forall r \ \in R$.

3. Calculate the $time\_conf$ of each item in $T(r)$ for rules $r \in R$ using Equ.1 and remove each item in $T(r)$ r such that $time\_conf < min\_time\_conf$. If $T(r) = \phi$, then exclude r from R.

4. Select $T_{time\_conf} = maximum\{T(r)\}$ and output remaining rules in R with their T_{time_conf}

T_{time_conf} value can be used as a window time $T_{registered}$ for systems such as CEP. 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 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 ordering internally.

3.3.1 Alarm Data Evaluation

We implemented our proposed method on top the RuleGrowth, which is a sequential association rule mining algorithm [26]. To evaluate our method, we used a historical alarm database from operational data of AVEA, one of the mobile telecom operators in Turkey with nearly 14 million customers as of 2013. AVEA's NOC receives up to

1 million alarms per day that are generated and transmitted from different network elements and IT infrastructure systems. Table 3 shows some of the most frequently occurring alarms. 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 (UMTS) and 6 OMC (Operation & Maintenance Center) elements. 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 is considered as one transaction c_i for our simulations.

Table 3: Some Frequently Occurring Alarms

ACDC FAULT
AIS RECEIVED
ALARMS FROM NETWORK ELEMENT NOT ARRIVING
BATTERY LOW
BTS FAULTY
CABINET OPEN
ETHERNET INTERFACE FAILURE
FAULT RATE MONITORING
INTERNAL LAN LINK BROKEN
LINK SET UNAVAILABLE
OSI SUBNETWORK INTERFACE OUT OF ORDER
RTCP SUPERVISION FAILURE
ROUTE SET UNAVAILABLE
SIGNALLING LINK OUT OF SERVICE

We set $min\_supp = 12$, $min\_conf = 0.8$ and $min\_time\_confidence = 0.9$, without loss of generality. Under this set-up, without $min\_time\_confidence$ value RuleGrowth algorithm finds 25 rules, whereas 9 rules were discovered with using the proposed method. We have a good match between the suggested rules and the feedback from the network support specialist. Fig. 9 shows the time difference histogram between the left and right sides of an alarm rule:

({AIS RECEIVED,
LINK SET UNAVAILABLE,
ROUTE SET UNAVAILABLE,
SIGNALLING LINK OUT OF SERVICE}
==>
{FAULT RATE MONITORING})

The x-axis is $time - confidence$ values in seconds, the y-axis is the number of occurrences of them for this rule. In this Figure 9, green columns represent the time interval values above the $min\_time\_confidence$ value and red columns represent the below ones. As can be observed from Fig. 9, in 15 transactions, an alarm type FAULT RATE MONITORING is observed exactly after $T_{time-confidence} = 1263$ seconds after the alarm types AIS RECEIVED, LINK SET UNAVAILABLE, ROUTE SET UNAVAILABLE, SIGNALLING LINK OUT OF SERVICE occurs. Therefore, for this example, the maximum occurrence of time difference value $T_{registered} = 1263$ seconds can be registered as the sliding-window time interval for the CEP system.

The experiments on the association rule queries can be run such as:

SELECT Alarm Correlation
FROM NOC Database
WHERE A,B,C,D $\Rightarrow$ E
WIN 1263 sec. **RETURN** BTS FAULT

which can be registered into the CEP engine. This query means that if alarm type A,B,C and D arrives in particular order, then alarm type E will happen within 1263

seconds and notify the network support specialist about the event BTS FAULT.

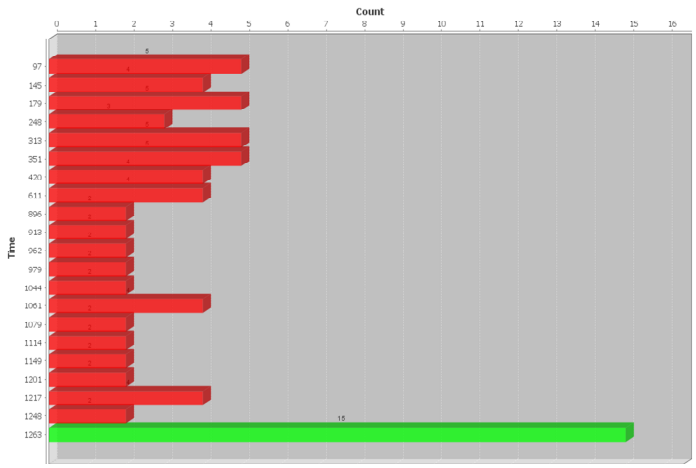


Figure 9: Time-confidence occurrence histogram for alarm rule ({AIS RECEIVED, LINK SET UNAVAILABLE, ROUTE SET UNAVAILABLE, SIGNALLING LINK OUT OF SERVICE} ==> {FAULT RATE MONITORING})

We implemented the framework presented in the previous sections in a Java-based prototype system. For the sequential association rule mining in the proposed method, we adapt the current sequential algorithm named RuleGrowth [26]. The prototype has a Java-based Graphical User Interface (GUI) that allows construction and execution of proposed method. The GUI is also used to adjust the parameters of the algorithms : minimum support, confidence and time-confidence. GUI can also show the execution time of the proposed method.

CHAPTER IV

CONCLUSIONS AND FUTURE WORK

In this thesis, two main contributions are discussed. Both of them summarize the high volume streaming data and present meaningful, actionable information to end users. The first one is finding "event correlations" over the data stream pairs on real GPS data of public transportation buses. Statistical correlation is described as a promising type of event detection and volume reduction technique over sensor streams. We used one type of sensor data and investigated one type of related application, which is real-time traffic and transportation management. The second one is alarm sequence rule mining, with a new parameter called "time confidence", that helps to automatically set time-window values for registered rules and also reduces the generated alarm rule count. A new alarm correlation method for mining sequential rules common to several alarm sequences is also presented. The proposed method has an extra time-confidence parameter that can both give confident time intervals and more effective rules into a CEP engine. This rule set with time confidence parameter can also be used with an event monitoring engine, such as CEP system. The simulations were performed to test the proposed method with real-world alarm data from AVEA. The evaluation of this method demonstrates the effectiveness of the proposed approach. The proposed method improves the accuracy of alarm rule finding space and can significantly reduce the number of alarm rules while exploiting the alarm correlation between different devices and network cards.

In this thesis, we concentrated on stream correlation and stream event sequence mining. For the stream correlation study, our future work will include methods for the correcting streaming data using regressions. Successful implementation of

the proposed CEP system will satisfy the real-time or near real-time data analysis requirements of data stream applications. While we were only concerned with wireless sensor data, the techniques discussed can be used with wired sensor data (e.g., border protection with sensor-equipped fences) and software-based (non-sensor) monitoring in financial, e-trade, computer network applications as well. For the stream event sequence mining, our future work will be integrating time-confidence value to the RuleGrowth [26] algorithm. Moreover, in this thesis, we concentrated on generating alarm correlation rules for each network element independently. As a future work, developing correlation with different network elements will be investigated.

Bibliography

[1] Wikipedia, "Data-stream management system." `http://en.wikipedia.org/wiki/Data-stream_management_system/`, 2010. [Online; accessed July-2013].

[2] EsperTech, "Event Stream Intelligence." `http://www.espertech.com/`. [Online; accessed July-2013].

[3] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: a new model and architecture for data stream management," *The VLDB Journal*, vol. 12, pp. 120–139, Aug. 2003.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, (New York, NY, USA), pp. 1–16, ACM, 2002.

[5] D. Patnaik, M. Marwah, R. Sharma, and N. Ramakrishnan, "Sustainable operation and management of data center chillers using temporal data mining," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, (New York, NY, USA), pp. 1305–1314, ACM, 2009.

[6] 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.

[7] "Apache Hadoop." `http://hadoop.apache.org/`. [Online; accessed July-2013].

[8] M. Wei, M. Liu, M. Li, D. Golovnya, E. A. Rundensteiner, and K. Claypool, "Supporting a spectrum of out-of-order event processing technologies: from aggressive to conservative methodologies," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, (New York, NY, USA), pp. 1031–1034, ACM, 2009.

[9] L. Golab and M. T. Özsu, "Issues in data stream management," *SIGMOD Rec.*, vol. 32, pp. 5–14, June 2003.

[10] A. Bulut and A. Singh, "Swat: hierarchical stream summarization in large networks," in *Data Engineering, 2003. Proceedings. 19th International Conference on*, pp. 303–314, 2003.

[11] S. Guha, C. Kim, and K. Shim, "Xwave: Optimal and approximate extended wavelets for streaming data," in *Proceedings of VLDB Conference*, 2004.

[12] T. Li, Q. Li, and S. Zhu, "A survey on wavelet applications in data mining," 2003.

[13] A. Arasu, S. Babu, and J. Widom, "The cql continuous query language: semantic foundations and query execution," *The VLDB Journal*, vol. 15, pp. 121–142, June 2006.

[14] E. Wu, Y. Diao, and S.Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, SIGMOD '06, (New York, NY, USA), pp. 407–418, ACM, 2006.

[15] Wikipedia, "Pearson product-moment correlation coefficient." `http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient/`, 2003. [Online; accessed July-2013].

[16] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: a review," *SIGMOD Rec.*, vol. 34, pp. 18–26, June 2005.

[17] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu, "Mining frequent patterns in data streams at multiple time granularities," 2002.

[18] N. Jiang and L. Gruenwald, "Research issues in data stream association rule mining," *SIGMOD Rec.*, vol. 35, pp. 14–19, Mar. 2006.

[19] S. Wallin, "Chasing a definition of alarm," *J Netw Syst Manage*, 2009.

[20] 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.

[21] 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.

[22] S. Laxman and P. S. Sastry, "A survey of temporal data mining," in *SADHANA, Academy Proceedings in Engineering Sciences*, 2006.

[23] 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.

[24] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of frequent episodes in event sequences," *Data Min. Knowl. Discov.*, vol. 1, pp. 259–289, Jan. 1997.

[25] 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.

[26] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, "Rulegrowth: mining sequential rules common to several sequences by pattern-growth," in *SAC'11*, pp. 956–961, 2011.

[27] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series," in *KDD'98*, pp. 16–22, 1998.

[28] 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.

[29] L. J. Fulop, G. Toth, R. Racz, J. Panczl, T. Gergely, A. Beszedes, and L. Farkas, "Survey on complex event processing and predictive analytics," technical report, 2010.

[30] 3GPP, "3gpp ts 32.111-2: Alarm integration reference point (IRP),," 2007.

[31] ITU-T, "Recommendation x.733: Information technology - open systems interconnection - systems management: Alarm reporting function," 1992.

[32] J. Pei, J. Han, and W. Wang, "Constraint-based sequential pattern mining: the pattern-growth methods," *J. Intell. Inf. Syst.*, vol. 28, no. 2, pp. 133–160, 2007.

[33] M. J. Zaki, "Spade: An efficient algorithm for mining frequent sequences," *Mach. Learn.*, vol. 42, pp. 31–60, Jan. 2001.

VITA

Ömer Faruk Çelebi is an MS student at Özyeğin University, Computer Science Deparment, İstanbul, Turkey since September 2010. Ömer Faruk Çelebi is currently working as an Information Intelligence Group Consultant at EMC. Previously, he worked as a Software Developer for Avea and Vodafone Turkey, mobile telecommunication operators, between December 2008 and July 2013. He received his BS degree from the Department of Electrical and Electronics Engineering at İstanbul Technical University, in December 2008. He also earned a business administration degree from Anadolu University, in Eskişehir, Turkey on September 2009. His research interests include CEP, data mining, cloud computing and software practices.