Department of Computer Science
Series of Publications A
Report A-2009-1

Data mining for telecommunications network log analysis

Kimmo Hätönen

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium CK112, Exactum, Gustaf Hällströmin katu 2b, on January 30th, 2009, at 12 o'clock.

University of Helsinki
Finland

Contact information

Postal address:
Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: toimisto@cs.Helsinki.FI

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911*

Telefax: +358 9 1915 1120

ISSN 1238-8645
ISBN 978-952-10-5195-1 (paperback)
ISBN 978-952-10-5196-8 (PDF)
URL: http://urn.fi/URN:ISBN:978-952-10-5196-8

Computing Reviews (1998) Classification: H.2.8, H.3.3, I.2.1, I.2.6

Helsinki 2009
Helsinki University Printing House

Data mining for telecommunications network log analysis

Kimmo Hätönen

Nokia Siemens Networks
P.O. Box 6 (Linnoitustie 6, 02600 Espoo)
FI-02022 Nokia Siemens Networks, Finland
Kimmo.Hatonen@NSN.COM

PhD Thesis, Series of Publications A, Report A-2009-1
Department of Computer Science, University of Helsinki, Finland
Helsinki, January 2009, 153 pages
ISSN 1238-8645
ISBN 978-952-10-5195-1 (paperback)
ISBN 978-952-10-5196-8 (PDF)
URL: http://urn.fi/URN:ISBN:978-952-10-5196-8

Abstract

Telecommunications network management is based on huge amounts of data that are continuously collected from elements and devices from all around the network. The data is monitored and analysed to provide information for decision making in all operation functions. Knowledge discovery and data mining methods can support fast-pace decision making in network operations.

In this thesis, I analyse decision making on different levels of network operations. I identify the requirements decision-making sets for knowledge discovery and data mining tools and methods, and I study resources that are available to them. I then propose two methods for augmenting and applying frequent sets to support everyday decision making. The proposed methods are Comprehensive Log Compression for log data summarisation and Queryable Log Compression for semantic compression of log data. Finally I suggest a model for a continuous knowledge discovery process and outline how it can be implemented and integrated to the existing network operations infrastructure.

Computing Reviews (1998) Categories and Subject Descriptors:

H.2.8	Database Applications [Database Management]: Data Mining
H.3.3	Information Storage and Retrieval: Information Search and Retrieval
I.2.1	Artificial Intelligence: Applications and Expert Systems
I.2.6	Artificial Intelligence: Learning

General Terms:
Algorithms, Design, Experimentation

Additional Key Words and Phrases:
Knowledge discovery process, Telecommunications, Frequent sets, Closed sets, Semantic compression, Inductive databases, Decision support systems

Acknowledgements

I am deeply grateful to my supervisors, Prof. Heikki Mannila and Prof. Hannu Toivonen. Heikki introduced me to an interesting area of data mining and sent me to this journey, whereas Hannu made me to reach the destination and finally finish the work. The thesis at hand has also benefitted from the insightful comments of Dr. Jaakko Hollmén and Dr. Jukka Teuhola and its language from gentle advices of MA Marina Kurtén.

I would like to thank my co-authors Dr. Mika Klemettinen, M.Sc. Markus Miettinen, Prof. Jean-François Boulicaut, M.Sc. Techn Perttu Halonen, Dr. Pekko Vehviläinen, M.Sc. Techn Pekka Kumpulainen and Prof. Risto Ritala, who have shared the painful process of clarifying and formulating ideas and results presented in this thesis.

This research has been carried out at Nokia Reseach Center, Nokia Networks and Nokia Siemens Networks, in each of which my acclaimed collegues have created the most inspiring atmosphere. I wish to express my gratitude especially to Dr. Olli Karonen and Mr Jouko Ahola who gave room for ideas, fostered the research to grow them up and saw that the ideas were able to mature and incarnate as a product. The financial support of Nokia Foundation and consortium on discovering knowledge with Inductive Queries (cInQ) is gratefully acknowledged.

The co-operation with colleques in industry and academy has been invaluable in searching for understanding of the challenges that the real world sets for data analysis. Great many thanks to you: Dr. Jukka Nurminen, Dr. Pirjo Moen, Prof. Erkki Oja, Dr. Albert Höglund, M.Sc. Tero Tuononen, M.Sc. Techn Antti Sorvari, Prof. Olli Simula, Dr. Jaana Laiho, Dr. Kimmo Raivio, Dr. Pasi Lehtimäki, Dr. Sampsa Laine, M.Sc. Techn Timo Similä, M.Sc. Mikko Kylväjä and all the others.

This long journey has contained many light and enthusiastic periods but sometimes it has lead through long and dark shadows. The support of my dear wife Tuija and the joy that our children Vili and Riia have brought have been the most important facilitators for this work. I also have to thank the teams P94 and KKI-35/40 of IF Gnistan that have kept me going.

Finally, I wish to thank my parents Hele and Ossi who lit the spark, set the example and who encouraged and supported me all the way with this dream that has come true.

Helsinki, January 2009, Kimmo Hätönen

Contents

Chapter 1

Introduction

Telecommunications network management requires rapid decision-making, which can be supported by data mining methods. The decision-making is based on information extracted from large amounts of data that are continuously collected from networks. What is required from data mining methods, how can they address the requirements, and how should they be integrated to the existing systems and operation processes?

The complexity of networks and the amount of monitoring data they provide are rapidly growing. The mobile telecommunications industry has rapidly developed during the end of the last millennium and the beginning of the new one. Twenty years ago mobile phones started to spread to all population groups in the western world. In 2007 about 1150 million mobile devices were sold [45]. All around the world operators are updating and developing their networks to increase their capacity and to facilitate new kinds of services.

Telecommunications network operation and management is based on the data that network elements provide [66, 119, 150]. The elements create log entries and alarms about events, system states and disturbancies and record time series about their performance. Network management systems collect data to the operation centers, where it is monitored and analysed to detect any defects or suboptimal states in performance or service quality. A middle-sized network can produce several thousands of alarms and tens of gigabytes of log and performance data per day. This data contains information about the performance of all network elements and services that the operator offers.

The volume of collected data sets a challenge for analysis methods and tools supporting network management tasks [44]. For example, how to recognise and identify immediately sudden problems that prevent large amounts of customer traffic, and how to find network regions and elements that require optimisation? These and many other data inspection problems are encountered continuously from day to another in network management processes.

Roughly at the same time with the growing popularity of mobile communications, in the 1990s, began a rapid and enthusiastic development in the research community: a new paradigm called data mining (DM) and knowledge discovery (KD) was developed [40, 54, 53]. This paradigm combines several research areas like databases, on-line data analysis, artificial intelligence, neural networks, machine learning and statistics, to name a few. Telecommunications systems that produce large amounts of data were among the first application domains of data mining methods [18, 59, 61]. Since then, many methodologies have been developed and applied to management and operation of telecommunications systems [98, 103, 108, 155].

Telecommunications network operation is a promising target for data mining applications. The network operation business consists of several activities, like network operation and management, customer care and billing, marketing, business management and so on [66, 119, 150]. These activities are closely related. They form a large dependency network where a change in one place affects everything else. Their management requires thorough understanding of network infrastucture, communication technologies, customers and their behaviour. New up-to-date information about networks and their development is needed all the time.

Knowledge discovery and data mining technologies have been applied in several related application areas like churn analysis [51, 94, 120], fraud detection and prevention [71, 10], network planning and optimisation [98, 104, 108, 155] and fault management [35, 58, 88, 146]. Most of the reported activities have been executed as separated data mining projects, whose results are then utilised in decision making. One of the greatest challenges for the knowledge discovery and data mining technologies seems to be how to support the continuous processes, like network maintenance where the same tasks are repeated day after day [104]. In these tasks the daily analysed data sets are large and time frames tight. They challenge the idea of iterative exploration, since there is no time for that.

Contributions of this thesis

In this thesis I study the application of data mining and knowledge discovery methods and processes in telecommunications network operations. The objective of the thesis is to find ways to assist operators in solving everyday problems in decision-making.

The thesis begins with an overview of the telecommunications network, its components and operation processes in Chapter 2. The chapter also reviews different types of data that the networks provide, especially different types of event data, which are the target data of this thesis. Chapter 3 discusses previous work on the knowledge discovery process and methods for monitoring telecommunications processes. It reviews definitions of frequent sets and their derivatives as a starting point for the contribution of this thesis.

The contribution of this thesis is threefold.

- Analysis of industrial requirements — Chapter 4

  I study industrial decision making as a target for data mining to support. First I analyse the decision making tasks and derive knowledge discovery tasks that support decision making. I also study the organisation and environment of a telecommunications operator to understand requirements they set for any new data analysis tool. This model clearly shows how some of the knowledge and assets like data mining expertise that is needed to apply data mining methods, are missing from the industrial environment. On the other hand, useful domain knowledge is often missed by method developers.

- Methods for industrial applications

  I introduce two methods that better address the operator environment needs in event log analysis.

  - Summarising and reducing analysed log data — Chapter 5
    The proposed method uses statistical features of the log entries to identify and remove repeating patterns from the logs. These patterns often form a large portion of the log volume, and their removal helps to make the log more comprehensible.
  - Reducing size of stored data — Chapter 6
    The summarisation method presented in the previous chapter is extended to compress data so that the original data can still be efficiently queried and analysed.

  The presented methods have been implemented and evaluated extensively. The summarisation method has been implemented in NetAct^{TM}Audit Trail — a security-log-monitoring tool of Nokia Siemens Networks.

- The knowledge discovery process in industrial environment — Chapter 7

  Based on the results of Chapters 4 – 6 and my experience with using them and integrating them into commercial products, I discuss how knowledge discovery can be integrated into industrial decision making in practice. I introduce a continuous knowledge discovery process motivated by the everyday decision making in network operations. I outline how this process can be implemented in the telecommunications operator information system and illustrate how the methods presented in this thesis integrate into an implementation.

Finally, Chapter 8 summarises the work and gives concluding remarks.

Contributions of the author

The contributions of this thesis have been published in several separate publications. The analysis of industrial decision making tasks (Chapter 4) is joint work with Tampere University of Technology [65]. In this, my contributions were the analysis and development of the model of knowledge discovery tasks and its application to telecommunications operation. The model of the industrial environment is based on joint work at the Nokia Research Center (NRC) and was first reported as publication [62]. My contribution was the creation and application of the model.

Chapters 5 and 6 are based on joint work at NRC and have been reported in two publications [55, 56]. My contribution in them was method development, validation and experimentation. In addition, I have greatly extended their experimental evaluation for this thesis. The results concerning the knowledge discovery process and its application to everyday decision tasks in industry (Chapter 7) are again joint work with Tampere University of Technology. Parts of the work have been reported in conferences [63, 156] or prepared as manuscript [65]. My contributions in these were the analysis and development of the knowledge discovery process and its application and integration to telecommunications network operations.

The research represented in this thesis has produced one filed patent application: "WO2003081433, Kimmo Hätönen and Markus Miettinen: Method and apparatus for compressing log record information." The summarisation method has also been implemented in NetAct^{TM}Audit Trail — a security-log-monitoring tool of Nokia Siemens Networks.

Chapter 2

Telecommunications networks

To understand the data mining needs of telecommunications monitoring tasks, we need to know how telecommunications networks are built up and operated. As an example I present the architecture of *Global System for Mobile communications* (GSM) networks [119, 67, 129, 152].

This chapter begins by a presentation of GSM network components and organisation in Section 2.1. After that I will introduce network operation processes and functions. I will discuss different monitoring data types, their properties and roles in Section 2.2. The section also shortly introduces test data sets used in evaluating the methods presented in this thesis.

2.1 Telecommunications network operation

2.1.1 Overview of the network

A telecommunications network provides mobile services for a large geographical area. This area can cover, for example, a whole country. A mobile-phone user can move around the area without losing his connection to the network. This is achieved by placing *Base Tranceiver Stations* (BTS) to give continuous coverage all over the area [119]. Figure 2.1 depicts a country that is covered by a network of some operator. On the left hand side of the figure there is a more detailed view that has been zoomed in to a small town. The view shows how BTSs have been distributed over the area.

A BTS has one or more transmitter-receiver pairs called *Tranceivers* (TRX). Tranceivers beam through BTS antennas, which are pointed to cover an area called a *cell* [155]. Figure 2.2 shows how the cells are placed around some of the BTSs of the network.

When a mobile phone makes a call, it creates a radio connection to a BTS. From the BTS the call is typically forwarded to a transmission network, which connects all the BTSs to other network elements and to outside networks like *Public Switched Telephone Networks* (PSTN) [119]. Figure 2.3 depicts a connection between two mobile phones as a black line connecting two BTSs that transfer the call. The dark grey lines between

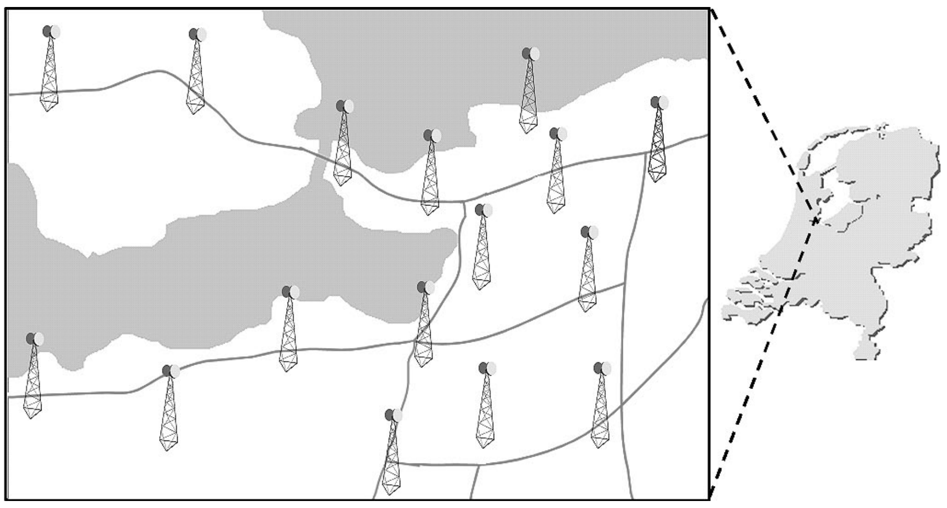


Figure 2.1: A country and a closer look at a small town, which is covered by a GSM network.

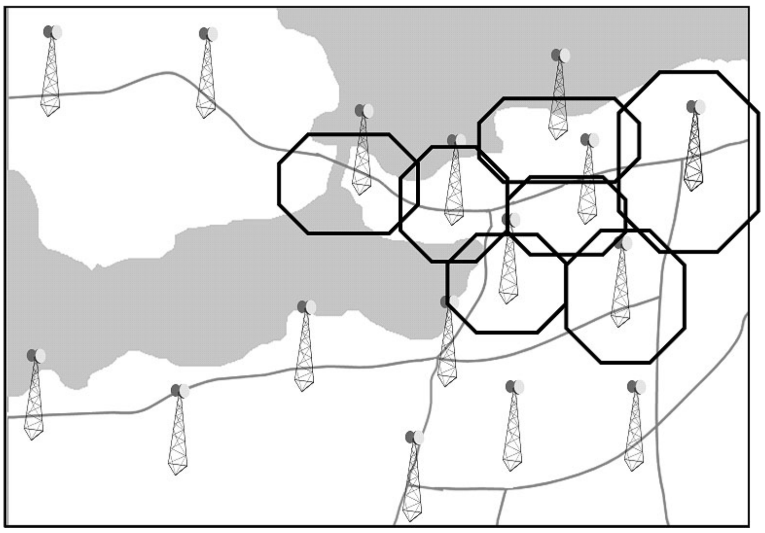


Figure 2.2: Cells of the network in a small town.

BTSs depicts the transmission network setup.

There are three separated subsystems in a GSM network architecture. These are the *Base Station SubSystem* (BSS), the *Network and Switching*

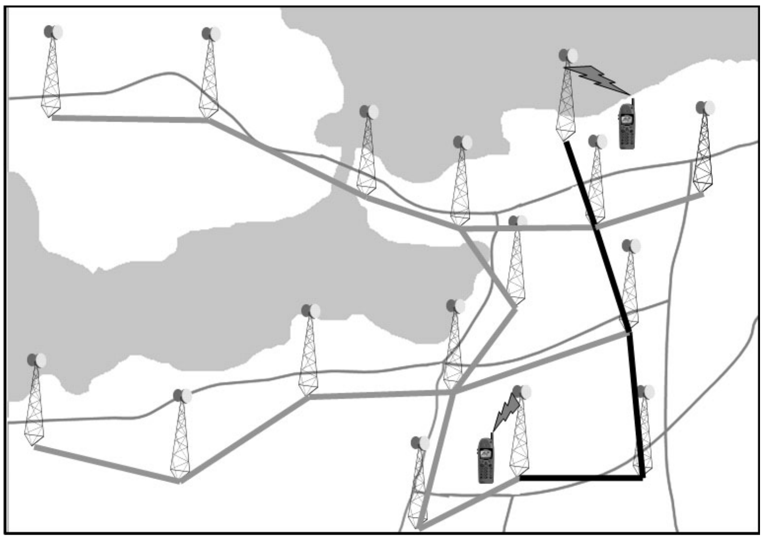


Figure 2.3: Call routing between two mobiles in the network.

SubSystem (NSS) and the *Operating SubSystem* (OSS) [119]. This subsystem division is depicted in Figure 2.4. The BSS is in charge of providing and managing transmission paths between the *Mobile Stations* (MS) and NSS machines, including in particular the management of the radio interface between mobile stations, like mobile phones, and the network. The NSS must manage the communications and connect mobile stations to the relevant networks or to other mobile stations. The MS, BSS and NSS form the operational part of the system, whereas the OSS provides means for the operator to control them.

As mentioned above, a BTS contains one or more TRXs [119]. Each BTS, in its turn, is controlled by a *Base Station Controller* (BSC), which can control several BTSs. For example, Figure 2.5 shows one BSC with BTSs associated to it. These BTSs have been grouped together, since they are close to each other, and the MSs move most often from one cell to another inside the BSC group.

BSCs in turn are grouped into a *Mobile Switching Center* (MSC) [119]. MSCs are the key elements of the NSS, whose main function is to coordinate the setting-up of calls and which is a bridge between the GSM network and an outside network such as a PSTN [129, 155]. The NSS also makes connections within the GSM network. Further elements of the NSS are*Visitor Location Register* (VLR), *Home Location Register* (HLR), *Authentication Center* (AuC), and *Equipment Identity Register* (EIR).

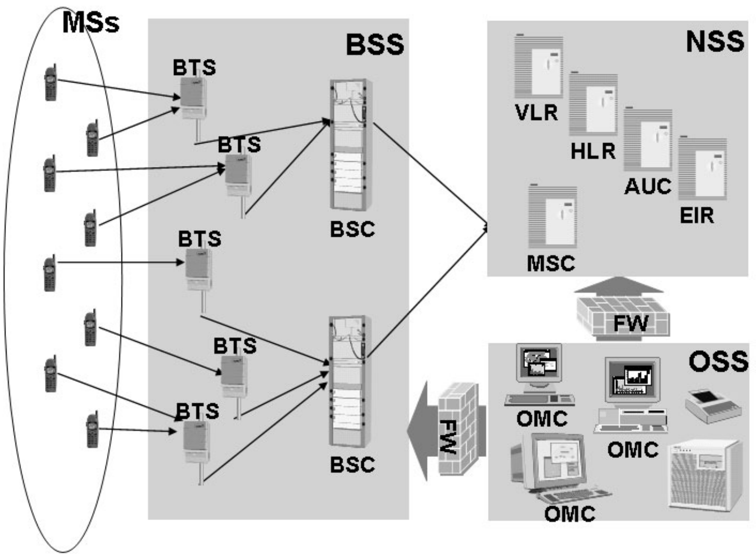


Figure 2.4: Subsystem division of a GSM architecture.

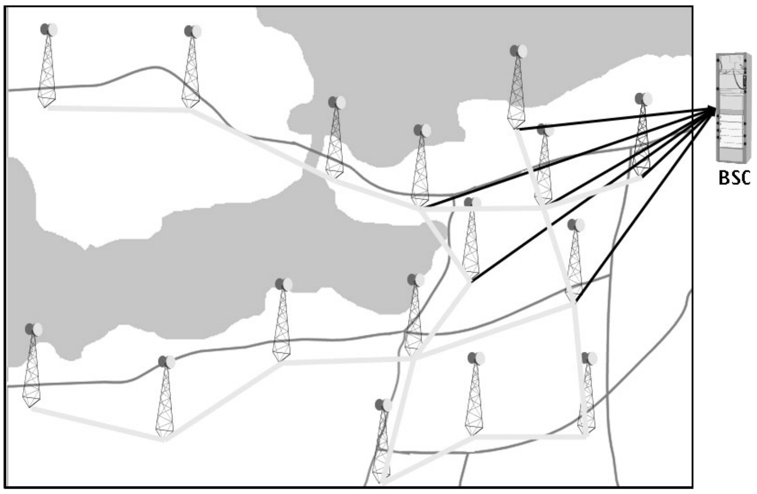


Figure 2.5: A BSC group.

The main tasks of the OSS are to operate and maintain the network and to manage both subscriber information and mobile stations [119]. The key element of the OSS is the *Operations and Maintenance Center* (OMC), which is used to operate the network by monitoring the network, managing

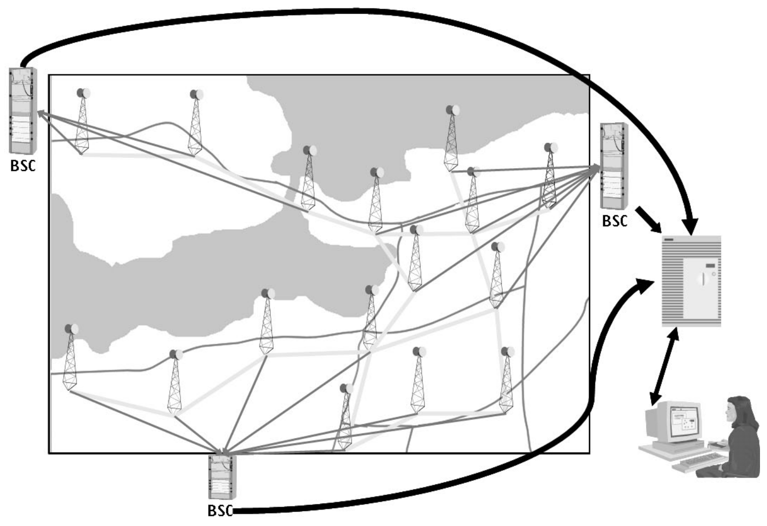


Figure 2.6: An OMC setup for BTS data transfer and monitoring.

changes in the network, maintaining the network stability and, whenever problems occur, to solve them [129, 155]. The OMC's main tasks are setup and change of network element parameters, monitoring the elements, and installation of software. In the heart of the OMC is a *network management system* (NMS) that is used to operate the network [100]. It is a software system, with which the operator personnel can monitor and access the network elements. An NMS is connected to the network's BSS and NSS via a *telecommunication management network* (TMN) [78]. Via TMN, the NMS acquires, transferes, stores and analyses alarms, measurement data and different types of network element logs. The NMS system servers are often protected with firewalls (FW).

The connection between BTSs and the OMC is depicted in Figure 2.6. The BSC collects the data from BTSs [119, 129]. From there the data is transferred to the OMC. As can be seen, data from several BSC groups is transferred to a database, which is then used for on-line and off-line analysis. The number of BTSs that are monitored at one OMC can be several thousands. Typically they are spread all over the area covered by the operator network. To be able to manage the whole network it is common that the network has been divided to *regional clusters* [79]. Each of these clusters is responsible for managing the network, for example, of a certain area as is depicted in Figure 2.7. In a large operator's organisation, the regional clusters can in turn be connected to a so-called *global cluster* as is

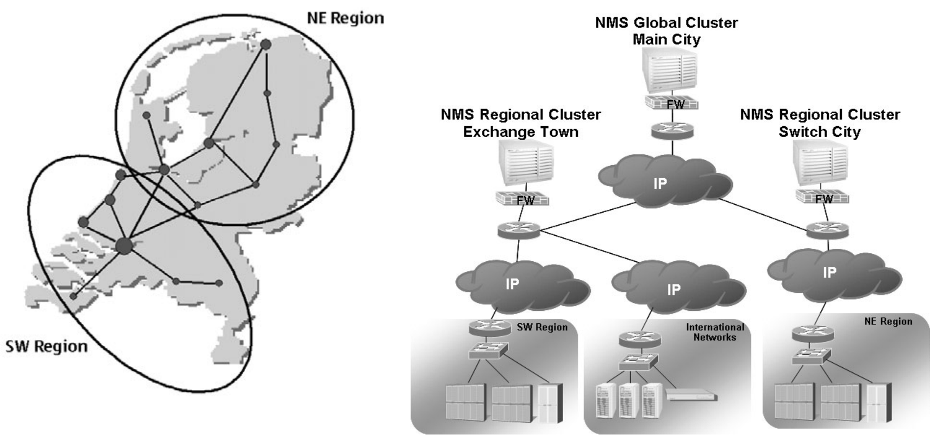


Figure 2.7: A cluster hierarchy of an operator business model and responsibility areas.

shown in the figure. Each operator divides responsibilities between regional and global clusters so that the division supports their own business model. Also the security solutions, for example, placement and configuration of firewalls, are operator specific [81].

2.1.2 Network management

In his thesis [155], Vehviläinen crystallises the telecommunications management network model [76, 151]:

> *The Telecommunications Management Network (TMN) model (Figure 2.8) is a collection of standards to provide a framework for managing the business of a telecommunications service provider. The model consists of four layers — business, service, network, and element management — which communicate information to one another. The purpose of the information transfer is to manage the telecommunications network. The model's top layers need information from the layers below to support decisions. Operational processes in each layer are documented in the Telecommunications Operations Map (TOM) [100, 149, 150].*

On the network management layer the TMN model identifies management functions. These functions are implemented in management pro-

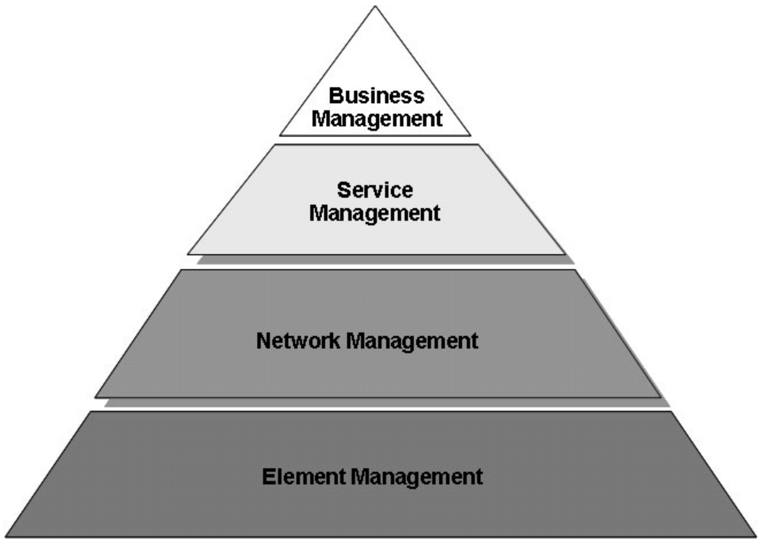


Figure 2.8: Network management infrastructure layers.

cesses. The network management functions can be grouped to three categories [119]: subscriber management, mobile station management and network maintenance. Subscriber and mobile station management share similar types of functions [66]. Examples of such functions are subscriber and equipment administration and subscriber and equipment tracing. Subscribers have to be activated or deactivated in HLR and their service profiles have to be downloaded and updated. Similarly, the EIR databases have to be administrated. Tracing of subscriber or equipment also share information and methods from both functions. In the case of stolen equipment or subscribers engaged in criminal activities, tracing of the affected handsets or subscriber identity module (SIM) cards has to be feasible throughout the network.

Charging administration is at the heart of the subscriber administration [119]. After each call, data, such as calling and called number, and time stamps, are recorded by the MSC and later sent to the billing system. In the case of some services, additional records may be generated, for example, by short message service (SMS) centres.

The network management is a larger set of functions [80]. It includes [66]:

- *Alarm handling and fault management; Indications of malfunctions or outages of any network component are transferred back to the NMS* *system. The technician can then remotely interact with the network component in question and try to repair the problem.*

- *Configuration management; Parameters necessary for network configuration such as frequency plans, next neighbour relationships or handover algorithms are sent from the NMS to all network elements. Configuration management also includes downloading of new software into the network and tracking software versions of all network elements.*

- *Performance management; All network elements generate a large variety of performance data. These data need to be collected by the NMS for further processing and analysis.*

- *Security management; This includes the handling of normal network security measures such as access control or system logs, but also the administration of GSM-specific security functions like authentication algorithms.*

From an end-user perspective, between him and the physical network implementing the requested service, there are several processes (see Figure 2.9) [100]. Together they aim to support the operator to deliver services, which are reasonably priced and well performed, so that the customer is kept satisfied. If any of these processes fails to provide other processes with information that they need or to maintain the part of the infrastructure that is under its supervision, the problem may propagate all through the organisation to the customer.

2.1.3 Network data

Network management is based on the data that the elements are generating [80, 66, 119, 129]. The NMS collects the data from BSS and NSS for further analysis and use on different functions. Each network management function handles data that it uses to generate information for consumption and use in maintenance processes.

The data can be divided into three categories: *system configuration data*, *system parameter data* and *dynamic data* that describes operation of network functions. System configuration data tells us how the network has been constructed and organised. For example, BTS locations and transmission network topology are included in such data. System parameter data defines how different system functions are implemented in the network and how they operate. Examples of such data are how BTSs have

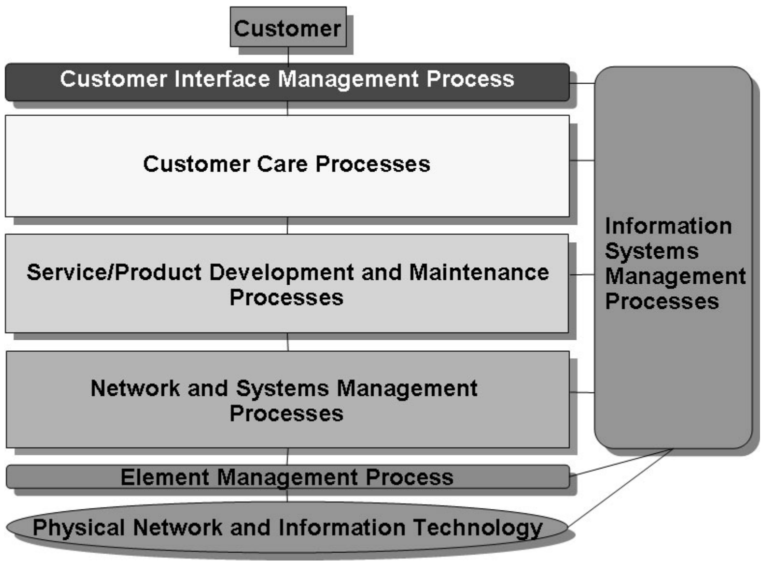

Customer
Customer Interface Management Process
Customer Care Processes
Service/Product Development and Maintenance Processes
Network and Systems Management Processes
Element Management Process
Physical Network and Information Technology
Information Systems Management Processes

Figure 2.9: Network management processes.

been grouped under BSCs and the BTS threshold for the number of accepted simultaneous connections. These parameters can either be adjusted by operator personnel or by autonomous processes that adapt to different traffic conditions in the network.

Dynamic data consist of measurement value and statistical time series, Call Detail Records, alarms and other events, system logs to name a few possible types of data. These describe the dynamic use of the network and processes implementing network functions.

The three given categories correspond to three description axes that are used to define GSM networks: static equipment view, static functional view and dynamic view [119]. The static equipment view shows the physical grouping of machines, the static functional view shows how network functions have been implemented in the physical network and the dynamic view describes the use of the network: how different functions are used and how they operate.

Subscriber and equipment management is based on the registers of known users and equipments [80, 66, 119]. These registers are updated when new users are added to the network or new equipment accesses the network. The user behaviour is recorded with *Call Detail Recods* (CDR), sometimes also called toll tickets [38, 119]. Every time a subscriber makes a call, the system creates a CDR. These CDRs are collected to a billing system and the system computes the value that can be charged from the subscriber. Charging of other services like MMS messages and data connections, is based on corresponding records. These records form the dynamic part of the information for subscriber and equipment management.

For performance management — network optimisation, maintenance, development and planning — the network elements count and record hundreds or thousands of measurement values or statistical time series [80, 148, 104, 152]. These time series are analysed in order to identify sub-optimal configurations, overload situations, radio interface quality and so on. An operator selects those time series, which describe the functions that are important in proper detail.

Configuration management handles the system parameter data [80]. Parameters are defined in planning software in the OMC and when the plan is considered ready, it is loaded to the network elements [104].

For alarm handling and fault management the system constantly monitors its functions and elements[80]. If the system recognises a malfunction or outage, it generates an alarm that is sent to the on-line operation room as an indication of a fault that should be fixed. The operator personnel can then remotely interact with the problem or send a technician to visit the alarm site.

The security function analyses access system logs to see whether the subscribers or operator personnel have misused their rights or whether there have been any intruders in the system [80]. These logs are recently augmented with logs provided by various security tools like virus scanners, firewalls and intrusion detection and prevention systems.

2.2 Telecommunications network data analysis

2.2.1 Event log analysis

A *log data* consists of *entries* that represent a specific condition or an event that has occurred somewhere in the system. The entries have several *fields*, which are also called *variables*. The entry structure might change over time from one entry to another, although some variables are common to all of them. Each variable has a domain of possible values. A small example fragment of log data is given in Figure 2.10. It is produced by CheckPoint's Firewall-1.

Firewall logs are a growing problem for operators [29, 28, 30, 43]. They can be huge. During one day, millions of lines might accumulate into a log file. Logs are used typically for two different purposes: either to find out why some connections or services do not work or to find out whether there

```
...
777;11May2000; 0:00:23;a_daemon;B1;12.12.123.12;tcp;;
778;11May2000; 0:00:31;a_daemon;B1;12.12.123.12;tcp;;
779;11May2000; 0:00:32;1234;B1;255.255.255.255;udp;;
780;11May2000; 0:00:38;1234;B2;255.255.255.255;udp;;
781;11May2000; 0:00:43;a_daemon;B1;12.12.123.12;tcp;;
782;11May2000; 0:00:51;a_daemon;B1;12.12.123.12;tcp;;
...
```

Figure 2.10: An example firewall log fragment.

are signs of security incidents. These signs can be monitored continuously day by day or they can be searched for on demand. In order to be able to track down incidents that have occurred a while ago, the logs have to be stored for quite a long time.

Firewall log entries may contain several tens of fields. Entries have date and time stamps specifying their creation time and they may be numbered with a unique entry id. They might also identify the firewall machine that actually created the entry. As an addition to these system information fields, entries contain information about the protocol used, source and destination addresses of the inspected packets, services used, users or processes involved and so on, i.e., everything that might affect the decision whether or not to let the packet pass through the firewall. For example, Figure 2.10 shows a small set of log entries. In the figure, only a subset of all possible fields is shown. Each entry contains an entry id, date and time stamp, the name of a destination service, a name and an IP address of the destination machine and the name of the protocol used, and finally one empty field.

When an expert needs to analyse firewall logs, he approximates the time range and selects values or strings that he assumes point to the information he needs. He starts to query them from the log database. This can be very laborious and slow, if the log files and database are huge. The query results easily become overwhelmingly large, when the selected query criteria are too wide. To focus on the essential data, the expert has to iterate with the query to find what corresponds to his information need. It may also happen that the query criteria are too strict or even totally misleading, and the answer does not contain any relevant data. Thus the expert has to reconsider the query and restart the exploration.

To decide whether the data respond to his information need, an expert has to check the found log entries by hand. He has to return to the original log file and iteratively check all those probably interesting entries and their surroundings. There are not many attacks that can be identified by one firewall log entry, but many that cause a known entry sequence pattern

```
...
11234 NE321    8.7.1997 020936 1234 2 Link_failure
11234 NE321/TRX1 8.7.1997 020936 5432 1 Call_channel_missing
11234 NE321/TRX3 8.7.1997 020937 5432 1 Call_channel_missing
11234 NE321/TRX1 8.7.1997 020937 6543 3 Link_access_failure
11234 NE321/TRX3 8.7.1997 020939 6543 3 Link_access_failure
11234 NE321/TRX2 8.7.1997 020940 6543 3 Link_access_failure
12345 NE123 8.7.1997 020942 8888 2 XXX/YYY:_alarm_indication_signal_received
12345 NE123 8.7.1997 020942 8888 2 XXX/YYY:_alarm_indication_signal_received
...
```

Figure 2.11: An example alarm log fragment.

in the log. Often, the most dangerous attacks are also unknown for an enterprise defense system. Therefore, if the data exploration is limited only to identified entry patterns, it may be impossible to find any new attacks.

In the network there are plenty of independent processes going on all the time. These processes emit alarms when they get disturbed by faults [119, 77, 58]. It often happens that many processes get affected simultaneously by a fault and they all start to send out alarms, not necessarily about the fault itself, but about its secondary reflections. Thus, the alarms and log entries that are sent actually carry second-hand information about the incident. They do not necessarily identify the primary fault at all.

Alarms that network processes emit are collected to some central monitoring applications [58]. This makes the analysis even more difficult, because at each monitoring application, there are alarms from several simultaneous sources merged in one stream. The volume of alarms flowing through the application grows and information is hidden under the masses. Connections between separate events — which are always difficult to identify — are lost, while the symptoms and reflection of separated problems merge into one information flow. For example, in Figure 2.11 there are alarms from two different network elements that probably have nothing to do with each other. The combined flow also contains noisy information caused by natural phenomena like thunderstorms or by normal maintenance operations.

A starting point for a network analyst in a fault situation is always localisation and isolation of the fault [119, 77], i.e., finding the area where the problem is located and identification of all network elements that are affected by the fault. Localisation and isolation are based on the assumption that it is probable that the fault itself is local although its reflections are widespread. In this situation alarms coming from the same network element or its direct neighbours are related to one reflection of the fault. After the localisation has been done it is easier to do the actual fault identification.

```
...
PRG1;1234;20040403;00:43:27;Module shutting down;
PRG1;1234;20040403;00:43:28;Start_operation received from element C1;
PRG2;3465;20040403;00:43:38;The Query application was started.;
PRG2;3456;20040403;00:43:40;Upload started;
PRG3;3678;20040403;00:43:52;The supervision application was started.;
PRG3;3678;20040403;00:43:57;The supervision application was ended.;
...
```

Figure 2.12: An example of an application log.

The alarm system can also be used to pass information about the normal operation [77]. A critical element can send notifications at regular intervals as a sign that it is up and running. If this heartbeat stops coming, the system can automatically start restoration of the element.

While alarm logs mostly contain signs of malfunctions and other problems, application logs record the normal system operation [43, 37]. As can be seen in Figure 2.12, Application logs are mostly created for debugging and maintenance purposes. They are analysed to identify the reasons for problems when they have been detected through alarms. In a normal situation, the same operations are repeated constantly from one day to another. Therefore, application logs contain a lot of redundant data patterns. All the signs of anomalous behaviour are buried under redundant normal entry patterns.

The same applies to the security logs. They contain a lot of redundant signs of normal behaviour and prevented incidents. If something anomalous happens, it is manifested in between normal entries. If the security monitoring system detects an unprevented incident, the operator needs to localise it and analyse what went wrong in prevention. There he needs to analyse not only security logs but also application logs to see what has been done in the network. Again, the user will benefit, if the redundant masses are removed and anomalous events are shown in all detail.

Alarms, application logs, system logs and security tool logs are the main application domain of the methods presented in this thesis. They all are lists of different entry types carrying information from network processes. As has been depicted with examples given in Figure 2.13, the entries can be divided into four categories based on the information that they carry. These categories are

1. Entries as signs of normal operation

2. Signs of prevented problems

3. Signs of known problems

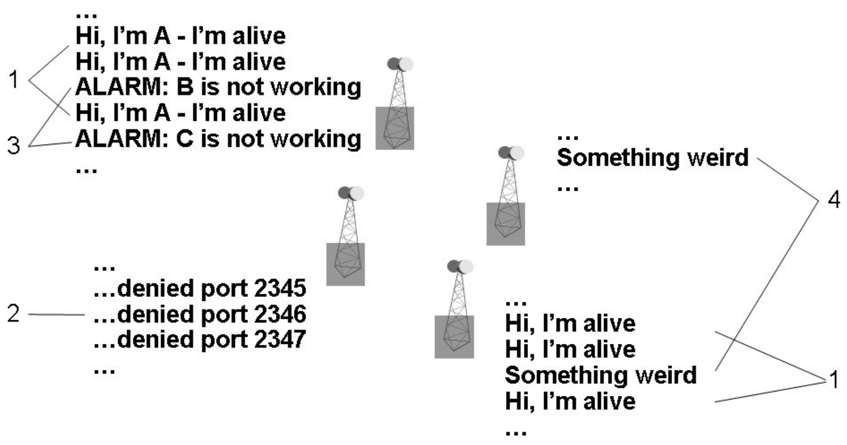


Figure 2.13: Different types of events marked with the numbers of the corresponding information categories.

4. Signs of previously unknown problems

Sometimes the categories of prevented problems and signs of normal operation are combined [28].

Event data analysis is a constantly repeated task. The events and logs are generated all the time that the network is operational. When the number of elements in BSS, NSS and OSS is growing, the amount of analysed data is increasing. Large pieces of data — especially those entries that are signs of normal operation — are repeating similarly from day to day. Also signs of problems — like the footprint of a prevented port scan depicted in Figure 2.13 — also contain large numbers of repetitive entries or entry sequences. The more valuable details, especially signs of hidden problems like intrusions, are hidden under these masses.

2.2.2 Data sets used in this thesis

In this thesis, several data sets from GSM networks have been used for testing and verifying the provided algorithms and their derivations. Data sets have been extracted from different types of networks and produced by different versions of network management systems and accompanied tools, during the ten-year period.

The set of data sets used in extensive experiments reported in this thesis include two sets of firewall logs. They were collected from two sources: a set of firewalls guarding an enterprise network and producing hundreds of thousands log entries per day, and a firewall standing between the Internet and a research network producing some thousands of entries per day. The time between collection of these data sets was four years. Each continuous subset covers a time period that varies from two weeks to some months. The same kind of firewall is used in commercial telecommunications networks in several places. A more detailed description of the test data sets is given in Section 5.4.4.

2.2.3 Performance data analysis

Telecommunications network monitoring is done with the data that is generated in the network elements (NEs). The elements count all the operations that they perform to establish a data or voice connection [148, 104]. These operations vary from voice connection or data context reservation attempts to handovers or connection shutdowns. The elements also monitor and record the validity of connections by recording detected error rates, the signal strengths used and other physical quantities describing connection quality. The counters and quality factors — called indicators from now on — are summed or averaged over a given time period. For monitoring purposes the time period length typically ranges from some minutes to some hours. As a result, elements provide a continuous series of values for each observed indicator. This time series data complements the alarm and log data. I introduce the performance data and its analysis briefly here but the detailed analysis methods are outside of the thesis scope.

Operators use the network performance data for several purposes including, for example, on-line trouble shooting, performance optimisation, and — for planning purposes — estimation of long-term development in traffic and service usage. For each operator and network the operators have their individual ways to judge whether the indicators in the network show that the cells are performing as expected and whether they are in their normal or optimal state [155].

The troubleshooting and network optimisation starts with detection of problematically behaving network elements, users or services. This can be done either by analysing the collected time series data and detecting values that are known to be outside the allowed value ranges or by detecting changes in the behaviour of network elements, users or processes. *Visualisation* of user behaviour or process states [69, 101, 157, 68] and KD methods like *anomaly detection* (or novelty or outlier detection) [70, 95, 97] or *unsupervised clustering* [134, 109, 101, 96] have all been used for detection of unwanted behaviour in telecommunications systems. These methods can be based, for example, on use of neural networks like *self-organising maps*

(SOM) [92], *decision trees* [19, 132] or *rough sets* [127]. These methods have been used to learn rules like indicator value range combinations that are typical for some problem situations [157, 105, 155, 99, 98, 103].

All the above-mentioned approaches need the scaling to find a task-specific balance between indicator value ranges [103]. In most cases the methods are very sensitive for scaling of data. Analyses can reveal totally different results depending on how the scaling has been done [63]. For example, the cluster analysis results can either emphasise the types of normal behaviour or reveal abnormal or problematic states of the process [64].

2.2.4 Knowledge discovery for network data analysis

As was discussed earlier, networks do effective self-monitoring and provide a lot of data in different forms about their behaviour. This data is the basis for all decision-making at all levels of their operation. The data contains information about the technical functioning of the network infrastructure, subscriber behaviour and use of services. An operator must know all these aspects well to be able to optimise its network and to gain maximal profit out of it.

The largest problem in many decision-making tasks, on all levels of the network operations, is the huge amount of data that needs to be analysed. The knowledge discovery process and data mining methods provide promising tools to assist in this. They are planned for analysis of large data masses that are available in some regular format. Telecommunications networks provide such data, which is collected to its operations and maintenance center, from which it can be transferred further to analysis systems.

For example, handling and analysis of alarm data has been a target of knowledge discovery. The number of alarms produced in a telecommunications network varies greatly, but typically there can be about $1000 - 10,000$ alarms a day, in the worst cases even more than $100,000$. The operations and maintenance center (OMC) of the telecommunications network management system receives all the alarms. OMC stores them in an alarm database, may filter them, but most importantly it displays the alarms to an operator, who then decides what has to be done with them. Analysis of the alarm flow is a difficult task, because

- The size of the networks and the diversity of alarm types mean that there are a lot of different situations that can occur.
- The alarms occur in bursts, and hence there is only little time for operators to decide what to do with each alarm. However, when a lot of alarms occur within a short time the operators should intervene.

- The hardware and software used in telecommunications networks develop quickly. As new elements are added to the network or old ones are updated, the characteristics of the alarm sequences are constantly changed.

To alleviate the problem of processing the alarms, alarm filtering and correlation [83, 82] was introduced to reduce the number of alarms that actually are shown to the operators and to raise the abstraction level of the information shown. Alarms are filtered at each level of the hierarchical network: a node sends forward only part of the alarms it receives from its subnodes. Alarm correlation means the combination and transformation of alarms so that several alarms are replaced by one alarm of better information value, which is sent further. Alarm filtering and correlation require stored knowledge about the processing of the alarm sequence.

Filtering and correlation serve to diminish the number of alarms that the operator sees. However, the alarm-handling software should ideally also be able to predict the behaviour of the network, i.e., to be able to warn the operator beforehand of severe faults. Such faults typically arise from or can be located in interconnected network component failures. While prediction of severe faults is a difficult task, the economic benefits that would be obtained from such predictions would be significant.

Another example of a growing problem in network data analysis where knowledge discovery methods can help, is the analysis and management of security related event data. Unlike alarms that have predefined type and classification taxonomies, security events may have a message text on some natural language like in most application logs, or have several fields of system parameter values like firewall logs, or are a combination of these two approaches like system logs. Event contents are application and system specific and targeted for engineers that are experts on the system.

Several security information and event management (SIEM) software and appliance systems have been developed recently [121]. They are large expert systems that store, parse, type, classify and correlate events in incoming event flow and alarm about detected known incidents. They use pre-defined rules often grouped by their functional modules. They typically have central server, to which all events from around the network are collected and where they are analysed.

Generating rules for alarm correlation engines [59, 60, 61, 58, 146, 35] and SIEM systems are a good example of what kind of problems knowledge discovery can be applied to. The knowledge discovery can also be used to support expert analysis work by summarising data and grouping elements by the data that they provide. It can offer tools for an expert to interactively explore the set of alarms or log entries, when manual fault or security incident identification and analysis is needed. Examples of resembling systems are found in a field of performance analysis, where knowledge discovery and data mining have been applied to radio network analysis [134, 101, 156, 99, 98, 103] and fraud detection in customer care process [71].

Chapter 3

Frequent pattern mining on logs

In this thesis I focus on applying knowledge discovery methodologies to analysis, use and storage of telecommunications log and event data. To facilitate that I present the knowledge discovery process in Section 3.1.

In data analysis, the thesis builds on frequent sets and episodes and their derivatives closed sets and episodes, which are defined in Section 3.2.

An early example of application of association and episode rules to network data analysis — the TASA system — is briefly described in Section 3.2.4. The lessons learned from it and other early industrial data mining trials are discussed in Section 3.4. As a conclusion the section outlines enhancements to TASA to support daily analysis of event logs better.

3.1 Knowledge discovery process

The purpose of Knowledge Discovery in Databases (KDD or KD in this thesis) is to find new knowledge in large data collections [90]. KD consists of consecutive tasks that are iterated to gain information to be translated into knowledge [90, 39, 53, 54, 161]. I follow the definition that the KD process consists of five steps: knowledge requirement setting, data selection, data mining, result interpretation, and knowledge incorporation [156]. Here the concept of data mining is limited to extraction of patterns or models from data.

In the Knowledge Requirement Setting task analysts together with management and domain experts set objectives for a knowledge discovery task. The objectives include requirements for information to be extracted from the data. *A priori* knowledge of the application domain is a prerequisite for specifying knowledge requirements [17]. While the KD proceeds, and more information has been extracted from the data, the knowledge requirements are further refined and expanded introducing changes in the settings of other tasks. If the requirements for the knowledge are vague, it is most probable that also the results of KD are not satisfactory.

The next step after data selection, the data mining process, described in Figure 3.1, is an iterative process itself. The black lines describe the constraints for selecting setup at different steps of the process and the grey

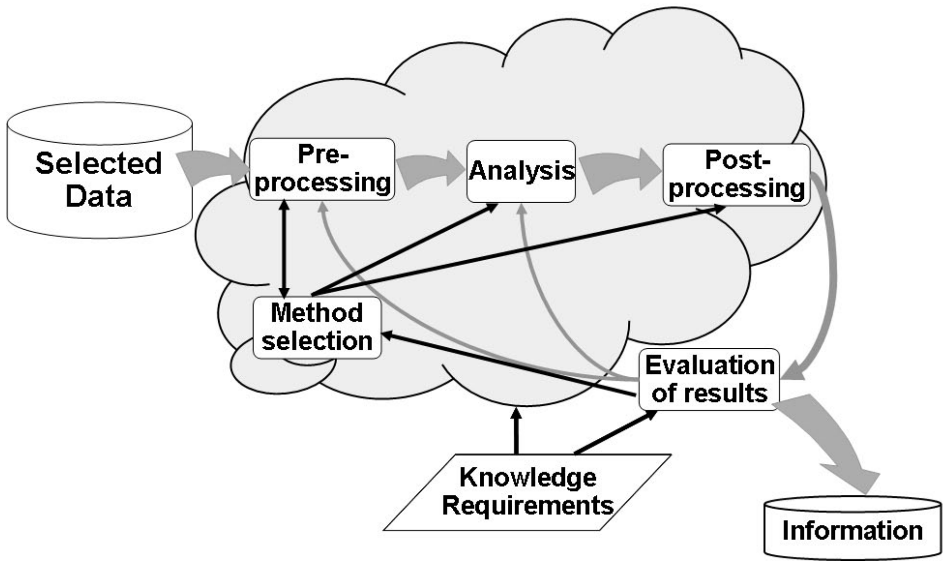


Figure 3.1: Structures of data mining process.

lines give the order of execution of the steps and flow of the data and information.

The data mining begins with preprocessing which reduces noise, copes with extremes, deals with missing values, and balances variables and their value ranges. Preprocessing is also an iterative process of analysing and modifying distributions of the data. The objective of the preprocessing phase is to enable the analysis methods to extract accurate and relevant information from the data [102].

Preprocessing is followed by the choice of analysis methods. The selection is based on statistical information and overall goals of the analysis task. After the methods are chosen, the desired features can be extracted. Feature extraction ensures that data is prepared in such a way that the selected analysis methods are able to provide the required knowledge from it.

The phase following the analysis consists of interpretation, presentation and validation of information and knowledge. The importance of user interface (UI) and form of results of the knowledge extraction system are emphasised. For a human analyst an understandable and plausible presentation is important. He must be supported in verifying the results against the data and domain knowledge. Only then the whole process has added value to decision making. The analyst has to verify whether the knowledge requirement has been completely met. Several iterations of the whole process are often needed before acceptable results are gained. Should it appear to be impossible to reach satisfying results, the knowledge requirement specification, selected method(s) or data set(s) have to be reviewed.

The knowledge discovery process was originally designed to support tasks, where a large and stable data collection that might contain valuable information is studied [17, 18]. Running the process requires strong domain and data mining expertise to succeed. It is iterative by definition and typically experts must make many selections based on the domain and data before they can achieve results that answer to the information need. Iteration takes time. As such the process is well suited to tasks, where history data is explored to find out information that helps operators to optimise the network or parameterise its subsystems. A good example of this kind of task is the alarm correlation example, introduced in Section 2.2.4.

In the following section, I will present the concepts that are in the focus of this thesis. After that, I will introduce the knowledge discovery tool Telecommunications Alarm Sequence Analyser (TASA) that was a starting point for this research. It is based on the presented concepts and uses them to study and develop data mining methods and systems for the knowledge discovery process for telecommunications data.

3.2 Frequent patterns

Frequent patterns are value or event type combinations that often occur together in the data. They provide information, which can be used to find rules or patterns of correlated or otherwise searched value combinations. A pattern is called frequent if the number of its occurrences in the data is larger than a given threshold.

In this thesis I discuss two kinds of frequent patterns: *frequent sets* [3, 115, 4] and *frequent episodes* [117]. Frequent sets consist of value combinations that occur inside data records like log entries. For example, with a frequency threshold 4 the firewall log sample in Figure 2.10 contains a frequent set *(a_daemon, B1, 12.12.123.12, tch)*. Frequent episodes, on the other hand, describe common value sequences like log message types that occur in the network. For example, in an alarm log sample in Figure 2.11 an expert can identify alarm sequence *(5432, 6543)* that occurs twice and is sent almost simultaneously by two TRX-elements attached to one base station.

3.2.1 Frequent sets

This section gives definitions for frequent sets in the context of the telecommunications network event logs [55].

Definition 3.1 (`Items`) `Items` is a finite set of *items* that are $field : value$ pairs, i.e., `Items`$= \{\texttt{A} : a_i, \texttt{B} : b_j, \texttt{C} : c_k, \ldots\}$, where $field$ is one of the fields in log entries and *value* attached to each field belongs to the domain of possible values of the $field$. □

Definition 3.2 (log entry) A *log entry* e is a subset of `Items` such that $\forall\ F : u, G : v \in e : F \neq G$. □

Definition 3.3 (log) A *log* $\mathbf{r}$ is a finite and non-empty multiset $\mathbf{r} = \{e_1, e_2, \ldots, e_n\}$ of log entries. □

In practice, log storage applications ensure the order of received log entries by numbering the log entries in arrival order and attaching the number to the entry. However, as the number field values are unique, they are generally not interesting in finding associations between log entry field values.

In a log entry, several fields may have overlapping value sets. For example, in a firewall log the *Source* and *Destination* fields may contain an IP address but the semantics of the values are different. The form of $field : value$ pairs makes it possible to include both IP addresses in the entries without the danger of mixing them up.

Frequent patterns are designed for symbolic data. However, numerical values can be included in the analysis if their value ranges have been discretised and they have been converted to categorial values.

Definition 3.4 (itemset) An *itemset* S is a subset of `Items`. □

The main properties that an itemset has with respect to a given log are a set of entries in which it *occurs*, i.e., of which it is subset, and the number of those entries.

Definition 3.5 (itemset support) A log entry e *supports* an itemset S if every item in S is included in e, i.e., $S \subseteq e$. The *support* (denoted $\text{supp}(S, \mathbf{r})$) of an itemset S is the multiset of all log entries of $\mathbf{r}$ that support S. Note that $\text{supp}(\emptyset, \mathbf{r}) = \mathbf{r}$. □

Definition 3.6 (itemset frequency) The absolute *frequency* of an itemset S in a log $\mathbf{r}$ is defined by $\text{freq}(S, \mathbf{r}) = |\text{supp}(S, \mathbf{r})|$ where $|.|$ denotes the cardinality of the multiset. □

```
Input: A log $\mathbf{r}$, frequency threshold $\gamma$
Output: The $\gamma$-frequent sets of $\mathbf{r}$
  1. $FE_1$ = the $\gamma$-frequent sets of size 1.
  2. For $(k = 2; FE_{k-1} \neq \emptyset; k{+}{+})$ do
  3.     $C_k$ = all itemsets of size $k$, all of whose $k$ subsets
             of size $k - 1$ are $\gamma$-frequent.
  4.     For all itemsets $c \in C_k$ do
  5.         $c.count = |\{e \in \mathbf{r} \mid c \subseteq e\}|$
  6.     $FE_k = \{c \in C_k \mid c.count \geq \gamma\}$
  7. od
  8. Output $\bigcup_{1 \leq i \leq k} FE_i$
```

Figure 3.2: Apriori algorithm for computing itemset frequencies.

Constraints are used to define interesting patterns. What is interesting depends on the task at hand, data and already available information. Originally only minimum frequency was used for this purpose [3, 115, 116]; later several other constraints have also been studied [16, 11, 133].

Definition 3.7 (itemset constraint) If $\mathcal{T}$ denotes the set of all logs and $2^{\texttt{Items}}$ the set of all itemsets, an *itemset constraint* $\mathcal{C}$ is a predicate over $2^{\texttt{Items}} \times \mathcal{T}$. An itemset $S \in 2^{\texttt{Items}}$ *satisfies* constraint $\mathcal{C}$ in log $\mathbf{r} \in \mathcal{T}$ iff $\mathcal{C}(S, \mathbf{r}) = true$. □

Definition 3.8 (minimum frequency) Given an itemset S, a log $\mathbf{r}$, and a *frequency threshold* $\gamma \in [1, |\mathbf{r}|]$, we define $\mathcal{C}_{\mathrm{minfreq}}(S, \mathbf{r}) \equiv \mathrm{freq}(S, \mathbf{r}) \geq \gamma$. Itemsets that satisfy *minimum frequency constraint* $\mathcal{C}_{\mathrm{minfreq}}$ are said to be *γ-frequent* or frequent in $\mathbf{r}$ and they are called *frequent sets.* □

The Apriori algorithm [4] (Figure 3.2) finds all the itemsets whose frequency is larger than the given frequency threshold. The algorithm is based on the observation that all subsets of a γ-frequent set are also γ-frequent. Therefore, the algorithm needs to study only those itemsets of size k, all of whose k subsets of size $k - 1$ are frequent. Such itemsets are *candidates* for the next frequency calculation round. The candidates are generated on Line 3.

3.2.2 Association rules

Frequent sets were originally developed for calculation of *association rules* [3]. An association rule $A \Rightarrow B$ describes association "*if A occurs in an* *entry then also B occurs in the entry*". A *confidence* of the association rule gives the observed probability *P("B occurs in an entry" | "A occurs in the entry").* The confidence of the rule is calculated from a data set $\mathbf{r}$ as $P(B|A) = \text{freq}(\{A, B\}, \mathbf{r})/\text{freq}(\{A\}, \mathbf{r})$.

A set of potentially interesting rules can be selected with *minimum confidence constraint.* According to it, rules with probability below the given confidence threshold are pruned.

3.2.3 Closed sets

A telecommunications network log can be seen as a sparse transactional database [15]. For example, in firewall logs fields potentially have a very large set of possible values, e.g., the value of the *Destination* field that contains the requested destination address, can be any IP address in the Internet. However, probably in most of the entries, the field contains addresses of those servers in an intranet, which are hosting services like web and mail that are open to the Internet. In practice many fields are very dense; they have only a few values from which one or a few are much more common than the others. This means that the encountered field value frequencies follow a very skewed distribution.

There are also lots of local correlations between field values. A high correlation together with the skewed exponential value distribution cause the number of frequent sets to increase dramatically compared to more evenly distributed data.

The Apriori algorithm works fine when the number of candidates is not too large. In a sparse database, the number of candidates usually starts to decrease after the frequent sets of size two or three have been computed. With data like firewall logs, which are dense, this does not happen. On the contrary, when there are many local correlations between field values, the number of candidates and frequent sets starts to expand quickly. This problem of a large number of closely related frequent sets can be solved with so-called *closed sets* [126, 12], which can be used as a condensed representation of a set of frequent sets.

Definition 3.9 (itemset closure) The *closure* of an itemset S in log $\mathbf{r}$, denoted by $\texttt{closure}(S, \mathbf{r})$, is the largest (w.r.t. set inclusion) superset SS of S that has the same support as S, i.e., $\texttt{closure}(S, \mathbf{r}) = SS$ s.t. $S \subseteq SS$ and $\text{supp}(S, \mathbf{r}) = \text{supp}(SS, \mathbf{r})$ and there is no itemset T s.t. $SS \subset T$ and $\text{supp}(T, \mathbf{r}) = \text{supp}(SS, \mathbf{r})$. □

In other terms, the closure of S is the set of items that are common to all the log entries, which support S.

Definition 3.10 (closed set and closure constraint) An itemset S is a *closed set* if it is its own closure in $\mathbf{r}$, i.e., S satisfies the *closure constraint* $\mathcal{C}_{\text{close}}(S, \mathbf{r}) \equiv \texttt{closure}(S, \mathbf{r}) = S$. □

The collection of all closed frequent sets contains the same information as the collection of all frequent sets, in particular, the identities and frequencies. If we consider the equivalence class that groups all the itemsets that have the same closure and thus the same frequency, the closed sets are the maximal elements of each equivalence class. Closed sets can be used to regenerate efficiently the whole collection of frequent sets without scanning the data again [126, 12]. Conversely, if the whole frequent set collection is available, a simple postprocessing technique can be applied to compute only the frequent closed sets.

Closed sets can also be extracted directly from highly correlated and/or dense data, i.e., in contexts where the approaches that compute the whole frequent set collection are intractable [126, 12, 164, 128]. Several algorithms can compute efficiently the frequent closed sets [13, 14].

3.2.4 Frequent episodes

The notion of association rules was generalised for sequences by defining *episode rules* [117, 114]. Episode rule $A \Rightarrow B$ describes association "*if A occurs in a sequence also B occurs in the sequence*". The confidence of the episode rule gives a probability *P("B occurs in a sequence" | "A occurs in the sequence")* The probability can be computed from data by computing *frequent episodes*, which reveal items occurring close to each other in a sequence and correspond to frequent sets.

The sequences in the log domain consist of log event types — for example, alarm numbers or message texts — which are ordered by their recording or creation times. The patterns, the so-called *episodes*, are ordered or unordered sequences of entry types of log entries occurring within a time window of specified width.

In telecommunications management systems, event creation, transfer and logging mechanisms introduce variation to the order in which entries are inserted into the log. Therefore, it has proven in practice that the most useful analysis approach is to use no order at all or some canonical order between the types inside windows.

The Apriori algorithm for frequent sets (Figure 3.2) can be modified to compute frequent unordered episodes [114]. The modifications needed are minimal. Instead of a log with entries as input, the algorithm gets a set of log sequence windows.

3.3 Telecommunication Alarm Sequence Analyser (TASA)

The *Telecommunication Alarm Sequence Analyser (TASA)* system [59, 61, 60] was one of the first ever KD applications in industry [18]. It was built to support knowledge extraction from telecommunications network alarm databases. It extracted rules for incorporation into the expert system for alarm correlation and filtering. I describe findings that were made while the TASA system was applied to real-world problems. I also discuss experiences and possibilities for improvements in methods and technologies.

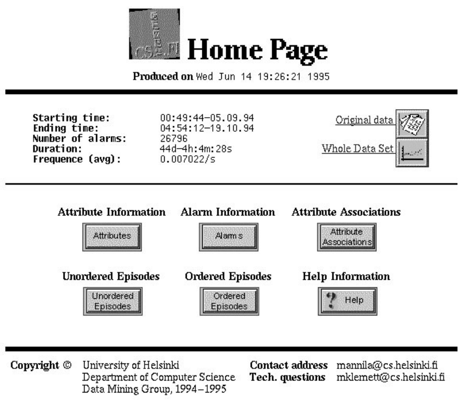
The purpose of TASA was to provide a data mining solution for analysing alarm sets collected from GSM networks. The objective was to find such alarm combinations that could be used in creating rules for correlation and filtering of related alarms and predicting forthcoming malfunctions.

The TASA system incorporated components for two parts of the KD process: pattern discovery and postprocessing. The knowledge discovered in TASA was expressed in terms of association and episode rules. The system located frequently occurring episodes from sequential alarm data with algorithms presented by Mannila and his collegues [117]. The algorithms were used to effectively find large pattern sets, typically thousands of rules. By finding a large rule collection, a large part of the iterative nature of the KD process was replaced by iteration in the rule postprocessing stage.

For postprocessing the TASA system offered a variety of selection and ordering criteria, and supported iterative retrieval from the knowledge discovered. The users could manipulate the discovered rule collection using selection and ordering operations, as well as more complex operations for including or excluding certain rule classes.

The user interface of the TASA system was implemented as HTML pages that were linked to each other. The pages provided interface forms for all the implemented selection and filtering primitives. They provided several views to the data, statistical figures and graphics describing it, and mining results so that the user could see how the data supported the extracted rule set.

The Home Page of a data set analysed with the TASA system (left-hand side of Figure 3.3) contains a brief overview of the most descriptive parameters of the data as a whole. These parameters include, for example, the time span of the data, number of alarms, average frequency of alarms, and so on. In addition, there are links to histograms that characterise the whole alarm data set as well as links to HTML pages that show the analy-



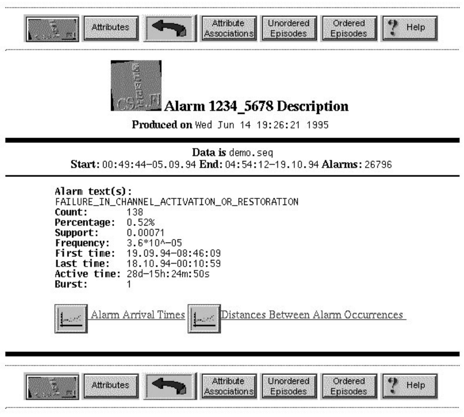
Home Page

Produced on Wed Jun 14 19:26:21 1995

```
Starting time:       00:49:44-05.09.94
Ending time:         04:54:12-19.10.94
Number of alarms:    26796
Duration:            44d-4h:4m:28s
Frequence (avg):     0.007022/s
```

Original data
Whole Data Set

Copyright © University of Helsinki, Department of Computer Science, Data Mining Group, 1994–1995
Contact address mannila@cs.helsinki.fi
Tech. questions mklemett@cs.helsinki.fi



Alarm 1234_5678 Description

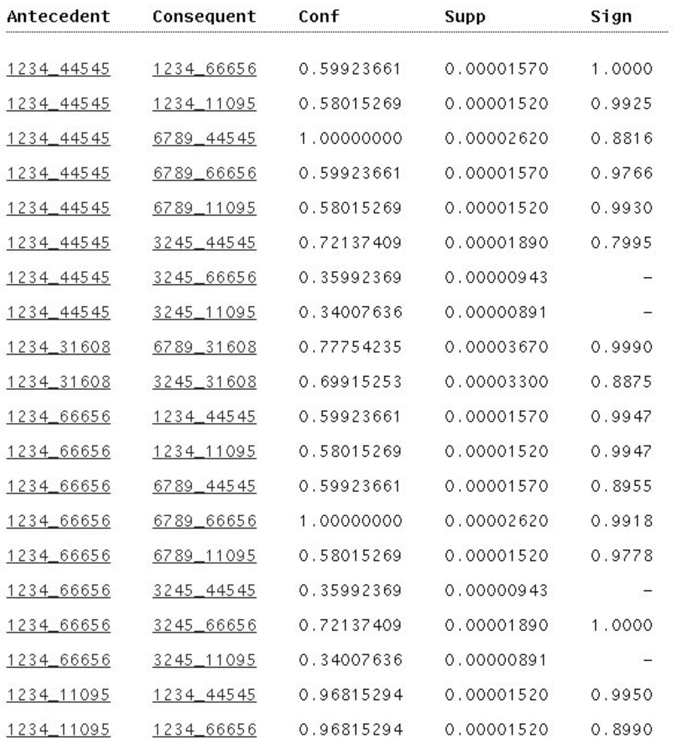
Produced on Wed Jun 14 19:26:21 1995

Data is demo.seq
Start: 00:49:44-05.09.94 **End:** 04:54:12-19.10.94 **Alarms:** 26796

```
Alarm text(s):
FAILURE_IN_CHANNEL_ACTIVATION_OR_RESTORATION
Count:       138
Percentage:  0.52%
Support:     0.00071
Frequency:   3.6*10^-05
First time:  19.09.94-08:46:09
Last time:   18.10.94-00:10:59
Active time: 28d-15h:24m:50s
Burst:       1
```

Alarm Arrival Times
Distances Between Alarm Occurrences

Figure 3.3: A home page of the TASA system and an alarm description page from the system.



Antecedent	Consequent	Conf	Supp	Sign
1234_44545	1234_66656	0.59923661	0.00001570	1.0000
1234_44545	1234_11095	0.58015269	0.00001520	0.9925
1234_44545	6789_44545	1.00000000	0.00002620	0.8816
1234_44545	6789_66656	0.59923661	0.00001570	0.9766
1234_44545	6789_11095	0.58015269	0.00001520	0.9930
1234_44545	3245_44545	0.72137409	0.00001890	0.7995
1234_44545	3245_66656	0.35992369	0.00000943	-
1234_44545	3245_11095	0.34007636	0.00000891	-
1234_31608	6789_31608	0.77754235	0.00003670	0.9990
1234_31608	3245_31608	0.69915253	0.00003300	0.8875
1234_66656	1234_44545	0.59923661	0.00001570	0.9947
1234_66656	1234_11095	0.58015269	0.00001520	0.9947
1234_66656	6789_44545	0.59923661	0.00001570	0.8955
1234_66656	6789_66656	1.00000000	0.00002620	0.9918
1234_66656	6789_11095	0.58015269	0.00001520	0.9778
1234_66656	3245_44545	0.35992369	0.00000943	-
1234_66656	3245_66656	0.72137409	0.00001890	1.0000
1234_66656	3245_11095	0.34007636	0.00000891	-
1234_11095	1234_44545	0.96815294	0.00001520	0.9950
1234_11095	1234_66656	0.96815294	0.00001520	0.8990

Figure 3.4: A selected list of episode rules from TASA.

sis results. Figure 3.3 also shows a basic description of the analysed alarm types included in the data. Figure 3.4 shows a list of rules related to the alarm type 1234 selected from a larger set of extracted rules.

3.4 Lessons learned

Research and development of TASA at the University of Helsinki were a starting point for the work reported in this thesis. TASA was later used at Nokia Research Center to analyse network log data collected from operational networks. Experiences with TASA proved the applicability and usefulness of the frequent pattern mining and knowledge discovery in general. The experiences also revealed limitations in TASA and assisted in identifying some problems in the methodologies.

3.4.1 Useful methodology

An alternative to gain more out of data Data mining and knowledge discovery methods are an alternative for operators to gain more out of their data [88]. Episode and association rules have been used in semi-automatic knowledge acquisition from alarm data in order to collect the required knowledge for knowledge-based systems like alarm correlators [61, 83, 82]. Given such rules derived from an alarm database, a fault management expert is able to verify whether the rules are useful or not. Some of the rules may reflect known causal connections, and some may be irrelevant, while some rules give new insight to the behaviour of the network elements. Selected rules can be used as a basis for correlation patterns for alarm correlation systems.

Iterative exploration A KD process includes a lot of iteration. Iteration is needed when searching for proper methods and their parameterisation to find the required information from the data. In TASA this means iteration with different thresholds and data selections. As a result the system reveals a set of selected informative rules.

As a by-product of the KD process experts can learn quite a lot from their data: “Which elements emit alarms?”, “What are the distributions of alarms types?”, “What are the most common alarm combinations?”, “Is there any correlation between the alarm sequences coming from different sources?”, and so on. This kind of information and knowledge about the network and its processes can be even more valuable than the rules found in the data. Such information can relatively easily be interpreted and connected to the experts’ mind maps about the domain.

3.4.2 Limitations of TASA

Large amounts of results The biggest problem with algorithms searching for rules is that they easily provide an overwhelming amount of them, especially if there are some alarms in the data that occur often throughout the studied time period. Together with more seldom but frequently occurring alarms they form plenty of rules where the more seldom occurring alarm type implies the more frequent alarm with high probability.

Another type of data that produces plenty of rules is such, where a group of alarms (for example, A, B, C) always occur together. In such a case the rule set contains a rule between each subset of the correlating alarm set (for example, $\{A => B$, $A => C$, $B => C$, $A => BC$, $B => AC$, $C => AB$, $AB => C$, $AC => B$, $BC => A\}$).

The network is large and lots of processes are going on in separate corners of it. The alarms from all the processes are collected to a few monitoring applications and signs of problems in different parts of the network are mixed. When episode rules are mined from this data, alarms from simultaneous but separate faults are correlated to each other. This correlation is statistically true, they occur during the same time period, but due to network structure, the faults causing these alarms probably have nothing to do with each other.

Interestingness and relevance of the results It is a tedious task to find interesting rules that reveal new information about the domain from the set of thousands or tens of thousands of rules. In many cases most of the rules found repeat the same newly found or already familiar information.

Different types of methods, like statistical descriptors or interactive browsing environments have been suggested in order to simplify identification of interesting rules [130, 74, 52, 144, 89, 131, 34].

The simplest way to reduce the amount of rules is to set thresholds higher. This restricts the search space. Clearly, the thresholds that can be used are not necessarily the ones that denote objective interestingness from the user point of view. Indeed, rare combinations can be extremely interesting. When considering previously unknown domains, explicit background knowledge is missing, e.g., the possible or reasonable values of attributes and their relationships.

Based on hands-on experience, simple-minded use of discovery algorithms poses problems with the amount of generated rules and their relevance. In the KD process it is often reasonable or even necessary to constrain the discovery using background knowledge. If no constraints are applied, the result set of, say, episode rules might become huge and contain mostly trivial, uninteresting or even impossible rules.

Intractable computation of frequent sets The execution of algorithms for finding frequent patterns easily becomes intractable. Algorithms try to overcome this by using effective methods to prune and limit the search space. Unfortunately, however, the log data contains a lot of redundant value combinations that make most of these algorithms reach their limits very soon. This happens especially when the interesting patterns are not those that occur most often in the data.

Fragmented information Association and episode rules show local correlations between data items. At their best, they provide small pieces which together — like pieces of a mosaic — form a big picture of the network and its processes. Unfortunately, when there are too many pieces and their relations are unclear, the big picture remains unclear or very fragmented.

3.4.3 Open problems

Lost environment information Many times, when the results of the TASA system were introduced to domain experts, the first question was "Where did these alarms come from and what happened around them?" The rule formalism extracts correlations, but at the same time it cuts off the connection between the correlation and the environment or situation where the occurrences of the rule were detected. It appeared that to be able to evaluate and validate any rule, experts needed to see as much information as possible about the network elements that were alarming. This information included not only alarms, but also measurements from the elements and their neighbours, geographical distribution of the elements sending the alarm, and so on. Even information about weather conditions were asked for.

Limits of knowledge discovery process model When the results of the research were applied to real-world systems, it became evident that also the knowledge discovery process that was identified in the research community was not applicable as such to the use of data mining methods in every-day work. Knowledge discovery is often defined [17, 39] as an iterative and interactive process, where a data set is analysed in the hopes to discover novel information to solve a real world problem [17]. This definition seemed to suit the separate academic and industrial data mining projects, where there are fixed data sets, resources with various types of expertise and a time frame from some months to some years. When the process itself is based on the iteration — or to put it in other words: on a trial and error principle — it takes too much time and effort to repeat it from the beginning day after day with new data sets and problems, especially when the main target and expertise of the people applying the methodology is to maintain an industrial process, not to develop new data mining technology.

Still, after more than ten years and several developed and applied data mining methods, there remains the challenge on how to assist operator personnel in their daily work. The networks are a rapidly changing and growing heterogeneous environment, where most of experts' time is spent in the routine tasks of searching and analysing faults and optimisation possibilities. In this environment, huge amounts of data are collected from the network every day, *a priori* knowledge tends to change, and the tasks have very tight time frames. This thesis focuses on developing methodology to bridge the gap between academic research results and industrial applications in this field.

Chapter 4

Industrial environment of data mining applications

Next, I will address decision tasks that are likely to arise in the operation of industrial processes, especially telecommunications networks [65]. I will identify decision tasks, study their properties and derive supporting knowledge discovery tasks. I will also characterise the users of the discovered knowledge and study the requirements they set for the knowledge and discovery tasks.

The development and application of data mining methods for industrial processes requires that they can be integrated to existing legacy software and used by the people that are experts in their own area, like network management, but who do not know much about data mining and analysis. In the latter half of this chapter, I will consider assets and resources that are available in industrial environments and derive requirements for data mining methods and tools [62]. I will also compare the environments of tool and method developers and users, to understand why some suggested solutions look fine in the laboratory, but do not succeed in everyday work.

4.1 Process information system

An operational telecommunications network can be seen as a service process, whose outcome are the communication connections between end users, or between end users and some services. Most decisions in a network operator organisation are based on the information collected from the network and stored in the process information system. The process information system acts on information and knowledge about the process state, process management, the system hosting the process, maintenance actions, and information system development. In a telecommunications network the data in the system consists of several registers, databases and log files as introduced in Chapter 2.

In general, an industrial process produces data through measurements about the volume or capacity of process functionalities, variations in raw materials, process state, services or products produced by the process (Fig-

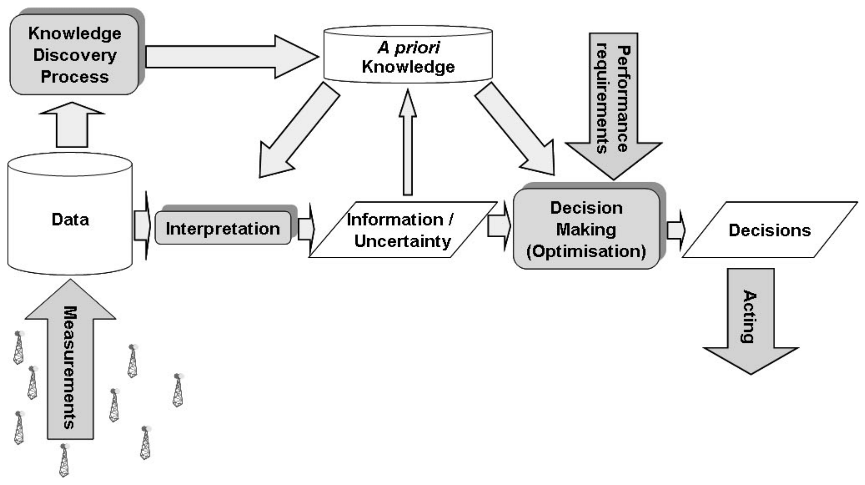


Figure 4.1: Process information system with assisting knowledge discovery process.

ure 4.1). By interpreting the measurements with *a priori* knowledge — often called background knowledge or domain knowledge in data mining literature — the process operator decides on the actions that optimise the process performance in terms of the utility of the process owner.

The *a priori* knowledge about the industrial process before it is in operational use ranges from general knowledge about the domain to detailed structural data. The general knowledge consists of several components, of which the most general are the so-called laws of nature, such as physical, chemical, and sociological facts about the process and its dynamics. Industry and regulators have agreed on standards and laws guiding the process implementation. Finally, manufacturers provide specifications about their implementation of industrial process structure, functions, their adjustments and management.

During the lifespan of the industrial process additional *a priori* knowledge is accumulated [125]. Traditionally this is based on expert insights of data and indicators derived from it. Separate knowledge discovery processes can also take place, for example, for better understanding of customers or some process states.

The *a priori* knowledge must be maintained. Maintenance actions are needed after either new industrial process setups or previously unknown process states are found during the operation of the processes. New services, products, process components, and technologies are taken into use during the lifespan. Unpredicted market trends may force changes on operational policies and service portfolios. As a result, some of the original *a priori* knowledge may become invalid and a need for generating the corresponding knowledge for the new environment arises and initiates a separate knowledge discovery process.

As Figure 4.1 suggests, the use of measurement data is two-fold. In addition to interpreting the data with *a priori* knowledge for operational decision support, the data can be used for accumulating knowledge for future use in knowledge discovery process. The information can be represented in forms ranging from mathematical models to unstructured natural language expressions [17]. The context and time span in which the *a priori* knowledge extracted from the data can be applied are limited to similar environment and process states in which the data has been produced. For example, if the industrial process is producing good products with a certain process setting and configuration parameters, it does not guarantee that the product output quality is good after the process components have been updated to a newer version.

The applicability of past data is further limited by the assumptions made in the analysis of data. During the knowledge discovery process, for example, the expert must make plenty of choices and fix several parameters based on assumptions of the data and its origin. Therefore it is rather tedious in practice to maintain data-generated *a priori* knowledge — in particular, to detect when it is outdated — and to validate it in order to avoid erroneous decisions [17]. Obviously, if the time spent in doing the analysis is longer than the validity period, the information provided is useless.

Separation of knowledge discovery and data intepretation processes is often a non-optimal solution. The processes require similar resources and expertise. To avoid doubling the effort an integration of the processes would be needed. The integration should simultaneously benefit the decision making and the accumulation and maintenance of accurate *a priori* knowledge.

The results and process of any data mining or knowledge discovery task have to be adapted to the needs of the users. The users, or consumers of information can be categorised to automatic control functions, expert systems, human experts and decision support systems assisting the experts. Each such consumer sets its own requirements for the form of information and knowledge (Appendix A, Table A.1). The requirements also depend on the decision level at which the user operates. Decision time frames, goal settings, and possibilities and abilities to make subjective choices are different. The users prefer different knowledge formats and differ in their ability to estimate and handle incomplete or uncertain information. Automatic control functions and expert systems have an inbuilt data processing engine. Human experts require additional data analysis and processing systems to make use of the information that is in the data streams. For experts the analysis results must be translated back to semantic meanings through interpretation of results.

The decisions that automatic control functions and human experts make are based on subjective choices integrated to *a priori* knowledge. Subjective choices concern the attitude towards risks and values (ethics, environmental impact, societal issues) [118]. In the decisions about the system operation the values can be reflected in the goal setting. For example, instead of simply maximising expected economic benefits, the decision-maker optimises the process under subjectively chosen constraints about the risk related to the chosen action and on environmental values. Although goals have subjective components the decision making is rational and consistent if the goals and values do not change over time. Making the goals and values explicit promotes a good decision-making practice and consistent operation of the organisation.

4.2 Decision support model

In order to assist in a decision-making process, the decision support system provides a model by which to predict the evolvement of the system state under any set of allowable actions, given the present and earlier states. A schematic decision support model is shown in Figure 4.2. The formalism has been adopted from Bayesian belief networks sometimes also called *probabilistic networks* [32]. The presented model is an abstraction that is used to illustrate the decision tasks and requirements that they set for supporting knowledge discovery. The model is not intended to be implemented as such. Due to the complexity of the information, many of its components cannot be easily expressed and computed. However, application of a related decision model to a limited set of every day decision tasks in paper industry has recently been studied further [136, 84]. On the field of *reinforcement learning* [86, 147], the presented model relates to *partially observable Markov decision processes* (POMDPs) [112, 1] that also could be studied as an option to implement the model.

The model consists of stochastic variables describing operator actions, system states, external trends and costs. The arcs in the model describe the conditional dependencies between the variables. The purpose of the

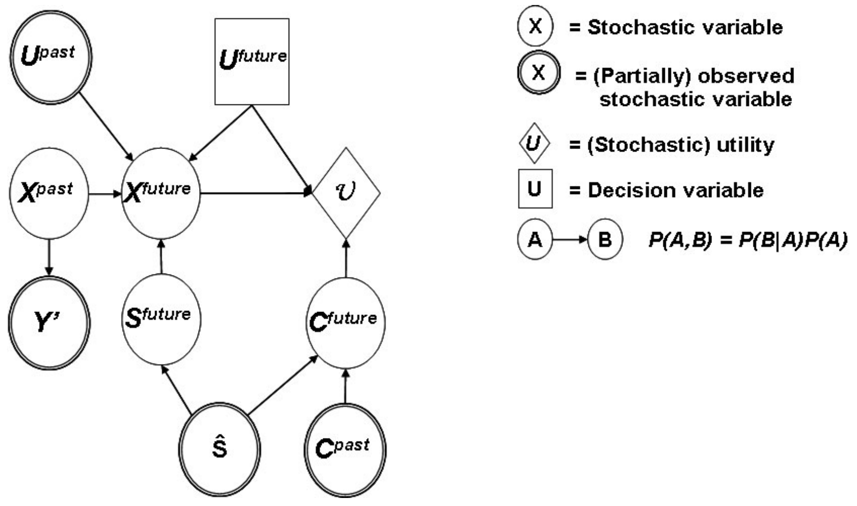


Figure 4.2: A schematic overview of a model for decision support system.

decision support system model is to identify and describe how observed or suspected changes in variables affect the other parts of the model. The knowledge discovery tools and methods are used to identify the current states and to support effect estimation.

The main task described in the model is to optimise the expectation of the utility function $\upsilon(U^{future}, X^{future}, C^{future}; d)$, where U^{future} is a set of actions chosen from the set of allowable actions, X^{future} the prediction of future system states, C^{future} the prediction of costs in the future and d a set of continuous and discrete fixed system parameters.

In Figure 4.2 past system states (X^{past}) and actions (U^{past}) affect system states in the future (X^{future}). It is important to notice that past system states are not observable as such for the support system but via measurements (Y') on them. The measurements do not give a complete picture about the state, furthermore they are uncertain. The description of the external world ($\hat{S}$) affects external scenarios (S^{future}) on process state evolvement, and cost evolvement. Observed cost histories (C^{past}) are the basis of estimating future costs (C^{future}). The decision making selects actions (U^{future}) from the set of possible actions. In optimal decision making the selected actions maximise the utility function υ under inequality constraints on future states X^{future}.

For example, in the telecommunications domain there are plenty of data sources, which provide information about the variables of the model. The alarm and measurement data collected from a network is a good example of measurements Y'. They provide information about the past system states X^{past}. Maintenance logs and trouble-shooting diaries provide information about the past actions U^{past}. The other sources for U^{past} are customer databases and external sources revealing information, e.g., about the number of sold mobile phones. The mass media and economical institutes doing studies on consumer behavior and needs provide the description of the external world $\hat{S}$. On the other hand, academic publications, standardisation organisations and manufacturers provide information about the external technology scenarios S^{future}. The costs in the past C^{past} are summarised several times a year from internal accounts to interim reports. Based on those figures and a description of the external world, the reports make estimations about future costs C^{future}. Together with selected actions U^{future} and the network infrastructure operator organisation, which are included in fixed system parameters d, all the information affects the utility function through some variables.

The analysis of decision-making above logically leads to four decision subtasks and to corresponding tasks for the knowledge discovery process implemented in decision support applications. The *knowledge extraction tasks* estimate and learn conditional probabilities based on stochastic variables given in model nodes. For these tasks, the source data is the measurements and the logged configuration changes and other actions in databases and system logs or in an external source data. The decision subtasks are *system state identification*, *system state prediction*, *cost estimation*, and *estimation of external actions*. The corresponding knowledge extraction tasks are to identify the conditional probabilities and to analyse how observed process conditions affect these probabilities.

Derived from Figure 4.2 the system state identification task can be defined as determining the conditional probability of system state, given the measured values: $P(X^{past}|Y')$ [137]. It defines how much of the past system state is known, given the measured values. The identification here means the ability to separate system states from each other, provision information about their properties and discovery of similar system states from the history data or system state description library. Fault identification and analysis, which is based on alarm logs, is an example of a system state identification task.

The conditional probability $P(X^{future}|U^{future}, U^{past}, X^{past}, S^{future})$ defines probabilities of possible system states in the future, given any allowable future actions, past actions, past system states and estimated external scenario. Thus it is the system model. The task of evaluating $P(X^{future}|U^{future}, U^{past}, X^{past}, S^{future})$ is called system state prediction.

For example, when an expert in an operation room recognises a set of critical alarms coming from the network, he has to identify the cause and its effects to the network, estimate the costs of the damage, and decide what needs to be done. One of the first things is to check whether there is any network management operation ongoing in elements giving the alarm and if not, has there recently been any and so on.

The task of cost estimation is to estimate the probability $P(C^{future}|C^{past}, \hat{S})$ that is the cost model. The cost model estimates probabilities of future costs by studying past costs and external scenarios. Estimation of external actions is simply the estimation of probability $P(S^{future}|\hat{S})$, in which external scenarios are related to the detected information about the external world. Both of these tasks require source information from external sources achieved, for example, by a business intelligence function of the operator, but also benefits from the analysis of user behavior inside the current network.

When all the four conditional probabilities are identified and thus the knowledge discovery tasks are completed, the operational decision-making under uncertainty can be formally expressed as a stochastic dynamic optimisation problem with, for example, the expected utility or the worst scenario utility (max-min) as the objective to be maximised [42]. Most of the knowledge discovery research has concentrated on supporting the system state prediction task. However, unless all the four tasks identified are covered, the decision support does not create value in the operation of the process (see [42] for a general discussion about value creation, and [85] for an industrial-specific case discussion). Obviously, some of the tasks can be left for the decision maker, but then this design decision must be made explicit.

4.3 Decision support on different operation levels

The three traditional levels in decision making have the following tasks. On the strategic level strategic scenarios are defined and the long term goals are set so that the industrial process structure can be designed, and the operational requirements and limits for tactical decisions can be set. On the tactical level the process performance with respect to operational requirements is optimised. On the automatic level the tactical decisions are implemented and their consequences monitored and controlled. The categories are somewhat overlapping and form a continuum [142, 46].

4.3.1 Strategic process decision support

On the strategic level the decisions concern the design of large and abstract entities and systems (Appendix A, Table A.1). A strategic decision aims to find the design of the operations (d), including the knowledge discovery and other information systems that will optimise the net present value (NPV, expected utility) of the process, subjected to scenarios about product portfolio, price and competition dynamics (C^{future}, $\hat{S}$). The net present value consists of accumulated and discounted revenues and costs over the lifespan of the industrial process. The two main strategic decision-making situations are the design of an entirely new industrial process, and a major restructuring of an existing one. The two cases differ in that during the restructuring some directly usable measurement information exists, whereas in the design phase of an entirely new industrial process, such as a telecommunications network, no measurement information exists.

The information, on which the strategic decision-making is based, comes from a multitude of sources and includes both abstract and intuitive components. Many of the sources are external and as such cannot be adjusted by the decision maker. The maximisation of NPV includes prediction of the future and the analysis of adaptivity of industrial process requirements.

For decision making on the strategic level many of the problems and scenarios are analysed in a conceptually or numerically simulated industrial process instead of a real one. Decision tasks may share some characteristics of similar existing operations, but the motivations and mechanisms are case specific anyhow. In strategic decisions not only the scenarios but also their frequency of occurrence must be specified. As a result the risks and uncertainty are quite high in strategic decision making.

Systematic strategic decision making must address the optimised tactical decision making under the scenarios defined, i.e., find optimal U^{future} for fixed C^{future}, $\hat{S}$ for each d. Strategic decisions aim to optimise performance over the long term. The knowledge discovery analysis tasks for strategic state identification and state prediction must be performed with limited or no measurement data about the target.

In the cost estimation, information gathered from similar systems and market information sources is combined. Cost estimation is tightly connected to prediction of external trends. Prediction of external trends combines mostly information from external sources with the intuition of the management. These two tasks are not sensitive to changes in the system setup but rather to major changes in the external world, like market development, demography, conflicts, and natural catastrophes in the world.

The challenge in making knowledge discovery to support decisions at the strategic level is making the pieces of information, which are in multitude of forms, commensurate. Two parts of the organisation can record the same types of events differently or use different formulas for calculation of, for example, a performance value [113]. Furthermore, the meaning of the terms and names vary from one department to another. A human expert is quite good at interpreting information from various sources but the task, if made with a computer, needs to be designed very carefully and results have to be cross-validated with results obtained with different tools.

4.3.2 Tactical process decision support

The role of knowledge discovery in supporting tactical decision making is to provide information about the process and the surrounding world affecting it (Appendix A, Table A.1). Information needs vary over the industrial process life span. The vital decision tasks to be supported by knowledge discovery during the industrial process start-up are the system state identification and system state prediction. In the industrial process start-up phase data and information are gathered extensively for both tasks, and, as a result of the data gathering, we gain knowledge of how different control actions affect the industrial process states. For example, in the start-up phase of a telecommunications network, alarm and event logs are collected. From them the operator tries to figure out, what the critical alarms or alarm combinations are and what actions should be taken when those are detected. Simultaneously the operator has to learn what entries can be filtered out without losing too much information.

Cost estimations and profit scenarios defined during strategic decision making will also be verified against the gathered data and information retrieved from it during the start-up period. The start-up period of the industrial process ends when the process is in a stable production mode.

In an industrial process production mode, a decision has to be made every time a process setup, parameters, utility function, or products change. In the changed state the knowledge must be updated. The update task is similar to knowledge discovery tasks during the start-up phase, when the operator learns characteristics and causes of the change. The needed updates mainly concern knowledge about the system state identification and system state prediction. Those parts of the knowledge that need to be re-evaluated depend on the nature of the change and how it affects the system. For example, if changes in system parameterisation cause the system to evolve into a state that is already known, only the prediction probabilities are affected. This may be the case, for example, when a new version of a software is loaded into the system. On the other hand, if the system enters an unknown state then both the state identification knowledge and the estimation probabilities must be updated. This might happen, for example, when there is a new bug in the updated software.

At the tactical level the external actions are usually estimated only locally with the granularity corresponding to the responsibilities of the supported decision maker. The granularity of the knowledge discovery tasks is typically limited to support tasks and responsibilities according to the structure of the operating organisation. The view to data probably differs if we are optimising a radio frequency plan of a small geographical area or ensuring the quality of end-to-end connections that a user experiences when moving around the entire country-wide network. However, organisational structures are rapidly changing and there is a real threat that a large part of knowledge, which is granularised according to organisational structure, becomes irrelevant in the reorganisation.

The data has to support the knowledge discovery task. As the definitions of decision tasks in Section 4.2 indicate, the available data for knowledge discovery must contain — in addition to measurements — a log of performed actions, history of system parameters and settings, estimation about future scenarios, and a definition of the set of allowable future actions. In many systems, unfortunately, only measurement value time series are available for knowledge discovery.

The system state identification and system state prediction tasks rely totally on the measurements and logs. If they do not contain information about the system state, it is impossible for the knowledge discovery process to generate it. On the other hand, after a change in the measurement set or in the measurement semantics, the gathered knowledge must be re-evaluated and validated.

4.3.3 Automated process decisions

The automatic control functions monitor incoming data flow and, if the defined antecedents apply, take actions that have been defined to be a corresponding consequence (Appendix A, Table A.1). The actions can either be active functional changes of, for example, the configuration parameters of the system, or they can be passive, in which the function identifies the industrial process state and signals results of its deductions to operator personnel.

For an operative automatic industrial process the quality of automated decisions has to be very high — there is no room for wrong decisions that cause the industrial process to fail. The decision tasks vary, starting from very fast optimisation with a decision period of milliseconds, for example, a decision when the system has to do handover for a call, and ending up in error detection and analysis tasks lasting days. Supporting knowledge discovery tasks can be classified as system state identification and system state prediction tasks. Applications making automated decisions solve either a high number of frequently occurring simple problems or very difficult and tedious problems requiring a lot of expertise [123].

When an implementation decision of an automatic control function is made, the following elementary questions must be considered: is it clear what the validity scope of the knowledge is, how fast the knowledge will be outdated, is it possible to detect the problems with the knowledge, what is needed to maintain and upgrade the functions, and will resources be available for knowledge maintenance? The assumptions made when building the control models have to be easily available for the operator. Either the industrial process or the operating personnel have to check whether the assumptions are satisfied in a manner applicable to the task: the validity scope of the model must be checked and monitored frequently.

4.3.4 Tactical decision support in network operations

The traditional knowledge discovery process model has been designed with the strategic decision making in mind. The knowledge extraction is done off-line, results are evaluated carefully and analysis has been repeated if needed. The model can also be applied to support the off-line knowledge extraction for automated decision making. However, the model does not fit to on-line decision making on tactical level in network operations. There decision tasks have tight deadlines, tasks themselves are repeating and based on data that is continuously collected from the network (Appendix A, Table A.1). Therefore, in this thesis I am focusing on knowledge discovery for decision support in two cases:

- situations in which decisions about design and operational requirements are made at such a fast pace that strategic-level decisions and tactical decisions can no longer be separated, and
- tactical decision making when there is a clear distinction between the levels.

4.4 Users and resources of applications

In order to derive requirements for knowledge discovery tools and applications, we present a model that describes and contrasts a tool provider

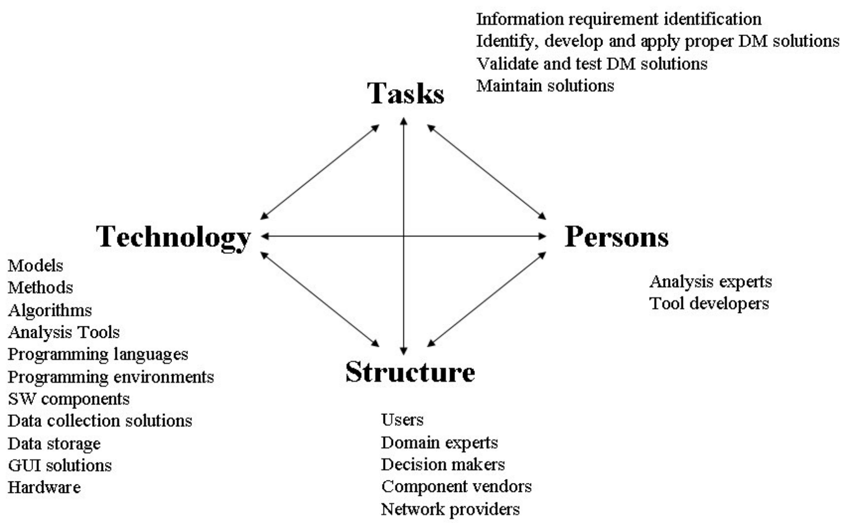


Figure 4.3: Interactions between data mining application issues from the developer perspective.

and a tool user environment in the telecommunications domain. To model these two environments we use Leavitt's diamond model [107]. It describes organisations as four interrelated components: tasks, technology, persons and structure, where structure represents the organisation as well as external stakeholders such as competitors. The interdependence between the different components of the model is strong. For example, changes in technology affect the way in which individuals relate themselves to the tasks they are responsible for and to the organisational structure. The model has been used as an analysis framework for, e.g., information systems [87], information system personnel and their roles [122], telecommuting [20], and telecommunications network planning tool implementation [124].

For this study the model has been divided into two views [124]: a developer view and a user view. The developer view illustrates a tool provider or developer organisation who selects whether DM methods are going to be used in tools. In the user view, a user organisation can be a telecommunications network operator or an IT department of an enterprise. The views are presented in Figures 4.3 and 4.4. The views and their comparison are used to derive the requirements for data mining tools.

Figure 4.3 shows the application of Leavitt's diamond model to development of the DM domain from the developer perspective. The tasks consist of requirements management, test data acquisition, method development,

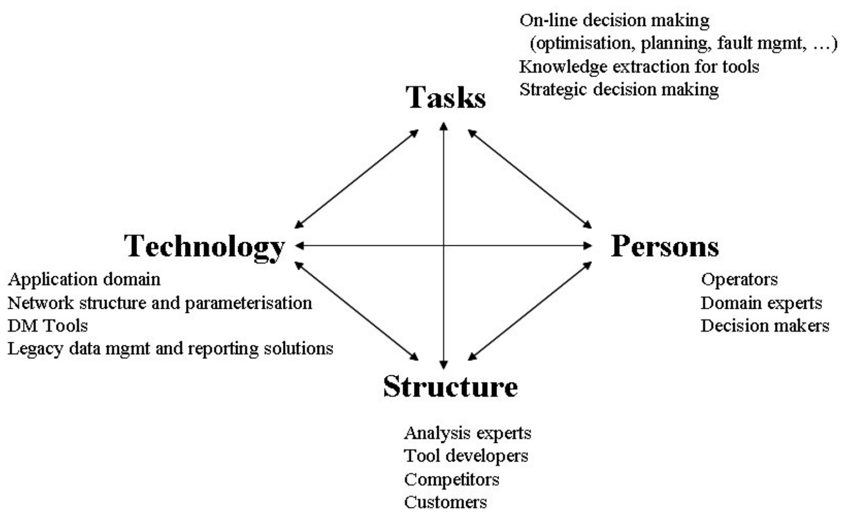

On-line decision making
(optimisation, planning, fault mgmt, ...)
Knowledge extraction for tools
Strategic decision making
Tasks
Technology
Application domain
Network structure and parameterisation
DM Tools
Legacy data mgmt and reporting solutions
Persons
Operators
Domain experts
Decision makers
Structure
Analysis experts
Tool developers
Competitors
Customers

Figure 4.4: Interactions between data mining application issues from the user perspective.

method and tool verification and tool maintenance. The directly involved persons are analysis experts and tool developers. The technology consists of models, methods, algorithms, DM tools and environments, programming languages and environments, software components, data collection and storage solutions, legacy data management and reporting solutions, graphical user interface solutions and, finally, of the analysed network and hardware. The structure contains tool users, domain experts, decision makers, and software tool, component and platform vendors.

Figure 4.4 shows the model from the user perspective. The essential tasks — making decisions in different types of situations — are related to network operation and development. Such tasks include, for example, configuring a new network segment, optimising the services in some cells or fixing acute and critical faults. The technology component consists of numerous items of the application domain and the monitored network, its structure and parameterisation. Data mining methods are seen as technology embedded in domain-specific tools. From the user's perspective, these tools should be integrated to the legacy data management and reporting solutions that still offer the major functionality of the monitoring system. From the user perspective, the structure contains analysis experts, tool developers, customers and competitors.

These two contrasting views are interdependent. For example, the technology component of the user view is linked with the task component of the developer view as the developed methods and tools are the key results that are used by the users. Furthermore, the analysis experts and tool developers of the persons component in the developer view can be modeled to be in the structure component of the user view, and vice versa.

The non-trivial interdependence between the two views is a reason for conflicts since the needs of developers and users are contradictory [124]. For example, from the user point of view the tools should make the execution of simple network analysis tasks very fast whereas from the developer point of view the tools should be easy to implement and maintain.

The successful exploitation of DM tools requires understanding of the requirements set for the tools from the user point of view. If those requirements are not met, then the users very easily just do not use the new technology but stick with the existing solutions and their direct enhancements.

When users are selecting their tools, they set requirements for the possible candidates. For applications in industrial production these requirements are quite strict and technological excellence is only one aspect in the selection process. Other requirements are set for understandability, integrability, effectiveness, continuation of development, guaranteed support and maintenance, and so on. If these requirements are not met by the tool and its provider, the method or tool might be abandoned without a second look at its technological achievements.

In the diamond model of user environment, data mining tools are included in the technology component. To understand how other interrelated components of the model affect the acceptability of new technologies like data mining, we study the connections between them and the technology component in the user view. These are connections between

- technology and persons,
- technology and tasks, and
- technology and structure.

4.4.1 Technology and persons

Persons who use DM in operator organisations can be, e.g., technicians, top level domain experts or top managers with a lot of experience in the business. A common factor among all of them is that they are typically skilled in what they are doing, namely in running telecommunications networks. They probably do not know too much about statistics or data mining.

This sets a requirement for any proposed tool or method: it must provide results using the terminology and semantics of the application domain. For example, pure statistical figures without a good explanation about what causes them and what they tell an analyst are not necessarily understandable for a domain expert. In other words, the tool provider has to attach a semantic interpretation in application domain terms to each statistical figure used.

As observed, experts are willing to assist in the development and adopt a planning tool if it provides immediate and accurate results to them already during the development period [124]. This is most probably true also with any DM tool. This is essential, since without the domain knowledge that the experts provide, the developer is not able to do the needed semantic localisation of the tool to the application domain. If the method is easy to understand and provides accurate results, experts will use it to assist them in their daily tasks, to play around with it and provide the semantic connection by themselves. This will also require a good user interface for the method.

4.4.2 Technology and tasks

DM tools

In network operation there are plenty of different tasks with different time constraints. The most urgent task is to fix critical faults that disturb the communications of a large number of mobile phones. These faults are monitored and, if detected, analysed on-line 24 hours per day. Less critical faults are analysed in priority order based on daily fault and performance reports. Every now and then the operator personnel go through the whole network in order to detect cells that are not working optimally.

For all of the above-mentioned analysis tasks the operator has numerous monitoring and reporting tools that follow up different parts and functions of the network. Any DM tool is an enhancement for the existing tools. They should assist the persons in their tasks, which they are used to perform based on the information provided by the existing tools. These tools are typically tightly linked to different management applications, with which the operators tune and fix the network remotely. This setup requires proper input and output interfaces to the new enhancements, which have to be integrated to the existing infrastructure.

Network structure

The structure and parameterisation of the network evolves constantly. Quite a large number of cell configurations — e.g., one percent out of thousands of cells — are updated on a weekly basis. This sets a challenge for the personnel: the so-called normal or optimal value ranges of several indicators derived from a cell or a BSC group are constantly changing. These changes have to be identified from the measurement value series and verified against domain knowledge.

4.4.3 On-line exploration vs. off-line DM

The shortest decision-making loops have been automated. There are closed control loops that monitor one or more indicator time series and adjust process parameters as a response to the incoming data. For these control functions the DM applications can provide information about the effects of different traffic and configuration combinations. This information can be extracted off-line either from a history data set or a simulated laboratory data set.

Another natural target for support are strategic decisions, which are based on data and information in various formats coming from several different sources. Analysis of this information closely resembles a classical KD process, where also analysis experts are involved.

Probably the hardest target for decision support are the tactical and short-term strategic decisions, where the time to make the decision is limited, the problem occurs either very seldom or is totally new and for which no analysis expert is available. In these tasks the DM tools have to be so easy to use that a domain expert is able to quickly extract the necessary information by himself. There is no room for iteration or full-scale data exploration, but in spite of that the analysis has to be well focused and straightforward to use.

4.4.4 Technology and structure

Analysis experts

One of the most critical differences between the developer and user views is in the role of analysis experts. They are DM experts that develop the methods used. In the developer view they are in the persons component. This means that they are available inside the organisation and actively taking part in different tasks.

In the user view, analysis experts are in the structure component. They are not part of the organisation using the tools and methods but rather externals, probably personnel of a tool provider or some consulting company. This makes them temporary options for any continuous analysis task. They might be used for giving training in the roll-out phase of a tool, but later it is usually an expensive option to use constant consultations.

Legacy solutions and Competitors

A basis for all the operator organisation acquisitions is the amount of expected utility. The utility can be in a form of more effective operations and cost savings, improved product quality, new and impressive services and so on. If it is possible to manage the business with the old existing infrastructure and the expected advantage that could be gained with the new solutions is less than what is required to update the old system and maintain the new one, then the acquisition will not be made. For example, if updating the legacy solution would require re-programming some of the central building blocks of the existing system and thus re-testing and debugging of all the solutions depending on it, the expected utility gain has to be very large before the organisation is willing to consider taking the risk of updating the system.

One element in the structure component — competitors — is the source for the need to upgrade operation solutions. If competitors are able to achieve lower maintenance costs by using more efficient analysis tools, this probably drives the organisation towards considering them. Otherwise, if their running costs are higher than those of the competitors, it will mean losing profits in the long run.

4.5 Summary of requirements

With knowledge discovery tasks in mind, I derived requirements that the network operator organisation and infrastructure set for data mining and knowledge discovery tools and methods. Below is a list of identified requirements.

- Application domain terminology and semantics used in user interface (Sections 4.4.1 and 4.4.4)
- Immediate, accurate and understandable results (Section 4.4.1)
- Easy-to-use methods (Sections 4.4.1 and 4.4.4)

- Interfaces and integrability towards legacy tools (Section 4.4.2)
- Adaptability to process information (Section 4.4.2)
- Use of process information (Section 4.4.2)
- Efficiency and appropriate execution time (Section 4.4.3)
- Reduced iterations per task (Section 4.4.3)
- Easy to learn (Section 4.4.4)
- Increases efficiency of domain experts by reducing time spent per task (Section 4.4.4)

These requirements have been guidelines when we have developed the methods presented in the following chapters of this thesis. The most essential requirement can be summarised as a need to support domain experts in their own language.

The TASA system fulfilled some of these requirements. It provided understandable results in feasible time and was relatively easy to use. It also had an interface to legacy tools that provided the source data. However, it also had some drawbacks: it was not able to adapt to any other process information, the use of it required understanding of probabilities and statistical concepts, it had several tens of parameters, it required iteration and lacked ability to collect history information and adapt to it.

Chapter 5

Comprehensive Log Compression (CLC)

Log file analysis is a part of many system state identification sub-tasks of tactical decision-making in telecommunications network operation. It is often very tedious to do. In many cases the logs are huge and filled with constantly re-occuring entries or entry patterns.

When results of the TASA system (Section 3.2.4) were evaluated by domain experts, it appeared that an expert did not want to know all association rules of the instances of frequently occurring entries or entry patterns, but he rather wanted to filter the most frequent entries away. Frequent patterns capture information in these repetitive entries. The *Comprehensive log compression* (CLC) method uses frequent patterns to summarise and remove repetitive entries from log data [55]. This makes it easier for a human observer to analyse the log contents.

5.1 Overview of the method

Automated filtering of uninteresting log entries is used, for example, in alarm correlation. It is based on pre-defined knowledge about correlations and importance of events. Before the system is usable, the operator has to acquire and define this knowledge and also continuously maintain it. The TASA system was designed to support knowledge acquisition for this automated decision-making.

As was described in Section 3.2.4, frequent episodes and association rules can be used to assist in defining rules for an alarm-correlation engine. This is possible since the set of alarm types is well defined and known. Each type has a semantic interpretation in a network domain and an attached representative, like a number or a constant alarm text.

The same method can not be applied to the maintenance and security logs as such. In these logs, the set of event types is much larger, some of the event types are unknown and new types keep appearing, when new components are added to the network. In some cases, for example, in firewall data, it can be difficult or even impossible to attach a unique type

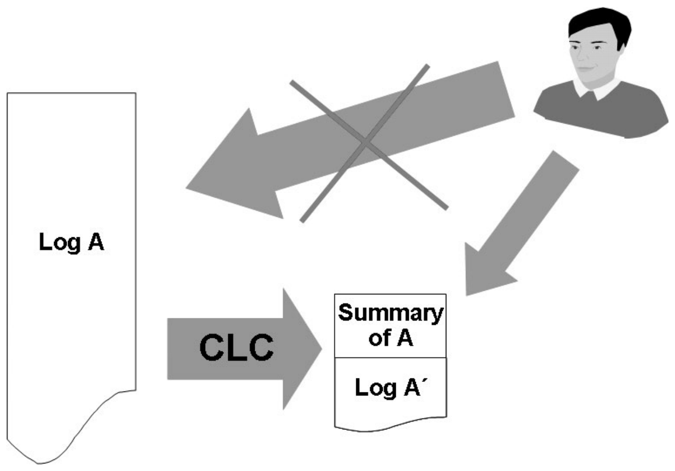


Figure 5.1: The CLC method creates a summary of a log and removes summarised entries.

to an entry.

The method that addresses this kind of changing data set, can not rely on a pre-defined knowledge base only. The maintenance of such knowledge would be too tedious. Instead, the method must rely on the data itself as much as possible. The method can not provide deep semantic interpretations since it is able to extract only statistical and syntactic information from data. With such information, however, the method can support a network expert to do the required semantic interpretations.

The CLC method dynamically characterises and compresses log data before it is shown to a human observer. The idea of the method is depicted in Figure 5.1. The CLC method analyses the data and identifies frequently occurring value or event type combinations by searching for frequent patterns from the data. The set of closed frequent patterns is presented as a summary of the log. Log entries that contain a closed set are hidden or removed from the log and only the remaining entries are shown to the user as a compressed log A'

The CLC method does not need any prior knowledge about the domain or the events. For example, no predefined patterns or value combinations are needed.

Making the summaries of the data benefits an expert in many ways. The frequent patterns included in the summary capture a lot of information. The information can include statistical factors — like how many entries contain the pattern — and descriptive factors — like when the first entry including the pattern appeared and how long these entries kept coming. It is also possible to compute information about the value distributions of those fields, which were not included in a pattern. This is useful, since when there is a massive burst of some event, for example, 100,000 entries, the operator is not interested in each individiul entry but the description of the whole set. On the other hand, when such bursts are removed from the log, other details, which were hidden between removed entries, become visible.

For example, beside user activity a firewall logs connections made by network elements and the network management system. In some firewalls most of the entries are caused by the system itself. Such entries typically contain plenty of repeating information with strongly correlated field values. Such entries can be effectively summarised by the CLC method. These summaries should reveal information, which is in line with other performance and log data collected from the network.

5.2 Formalisation

Definitions for frequent sets, closures and closed sets were given in Section 3.2.1. Those definitions have to be augmented with concepts that are needed for CLC. These concepts include notions of *coverage* and *perfectness* and the corresponding itemset constraints.

There are at least three possible measures that can be used to sort the patterns: *frequency*, i.e., on how many entries the pattern exists in a data set; *perfectness*, i.e., how perfectly the pattern covers its support set; and *coverage* of the pattern, i.e., how large a part of the database is covered by the pattern. Coverage balances the trade-off between patterns that are short but whose frequency is high and patterns that are long but whose frequency is lower.

Definition 5.1 (coverage) The *coverage* of an itemset S in a log $\mathbf{r}$ is defined by $\operatorname{cov}(S, \mathbf{r}) = \operatorname{freq}(S, \mathbf{r}) \cdot |S|$, where $|.|$ denotes the cardinality of the itemset S. □

Definition 5.2 (perfectness) The (relative) *perfectness* of an itemset S in a log $\mathbf{r}$ is defined by $\operatorname{perf}_r(S, \mathbf{r}) = \operatorname{cov}(S, \mathbf{r}) / \sum_{e \in \operatorname{supp}(S,\mathbf{r})} |e|$, where $|e|$ denotes the cardinality of log entry e. □

If the cardinality of all the log entries $e \in \mathbf{r}$ is constant $|e|$, then $\operatorname{perf}_r(S, \mathbf{r}) = \operatorname{cov}(S, \mathbf{r}) / (\operatorname{freq}(S, \mathbf{r}) \cdot |e|) = |S|/|e|$.

Definition 5.3 (minimum coverage) Given an itemset S, a log $\mathbf{r}$, and a coverage threshold $\kappa \in [1, \sum_{e \in \mathbf{r}} |e|]$, *the minimum coverage constraint* is defined as $\mathcal{C}_{\text{mincov}}(S, \mathbf{r}) \equiv \text{cov}(S, \mathbf{r}) \geq \kappa$. Itemsets that satisfy $\mathcal{C}_{\text{mincov}}$ are called *covering* in $\mathbf{r}$. $\square$

Definition 5.4 (minimum perfectness) Given an itemset S, a log $\mathbf{r}$, and a (relative) perfectness threshold $\pi_r \in [0, 1]$, *the minimum perfectness constraint* is defined as $\mathcal{C}_{\text{minperf}}(S, \mathbf{r}) \equiv \text{perf}_r(S, \mathbf{r}) \geq \pi_r$. Itemsets that satisfy $\mathcal{C}_{\text{minperf}}$ are called *perfect* in $\mathbf{r}$. $\square$

If the cardinality of all the log entries $e \in \mathbf{r}$ is constant $|e|$, then the minimum perfectness threshold can be specified as an absolute size of a pattern: $\pi \in \{0, 1, \ldots, |e|\}$, or as a difference from the entry size $\pi_d \in \{0, 1, \ldots, |e|\}$. Thus the minimum perfectness constraint can be equally defined as $\mathcal{C}_{\text{minperf}}(S, \mathbf{r}) \equiv |S| \geq \pi$, or $\mathcal{C}_{\text{minperf}}(S, \mathbf{r}) \equiv |S| \geq (|e| - \pi_d)$ correspondingly.

Definition 5.5 (filtering pattern) A closed set S is a *filtering pattern* if it satisfies constraint $\mathcal{C}_{\text{filt}}$, where $\mathcal{C}_{\text{filt}}(S, \mathbf{r}) \equiv \mathcal{C}_{\text{minfreq}}(S, \mathbf{r}) \wedge \mathcal{C}_{\text{mincov}}(S, \mathbf{r}) \wedge \mathcal{C}_{\text{minperf}}(S, \mathbf{r})$. $\square$

Definition 5.6 (summary) A *summary FF* of log $\mathbf{r}$ is the set of its filtering patterns, i.e., $FF = \{S \mid \mathcal{C}_{\text{filt}}(S, \mathbf{r})\}$. $\square$

An algorithm for finding the summary *FF* of log $\mathbf{r}$ is given in Figure 5.2. The algorithm first computes the collection of closed frequent sets *CFS* (Line 1) from the log $\mathbf{r}$. The summary *FF* is then the set of filtering patterns (Line 2) filtered from *CFS* by applying the constraint $\mathcal{C}_{\text{filt}}$.

If a filtering pattern S is a specialisation of a more general filtering pattern T, i.e., $T \subset S$ (Line 5), then the support of S is a subset of the support of T. Thus S would not filter out any entries that T would not already remove and S can be removed from the summary (Line 6). However, S is a specialisation of T and this relation could be recorded to be used by a tool to browse the summary.

The algorithm compresses the original log $\mathbf{r}$ by removing log entries that are in the support of any of the filtering patterns (Line 8). This is called *data reduction.* The algorithm outputs the summary *FF* and the uncovered entries of log $\mathbf{r}$, denoted with $\mathbf{r}_{\text{uncovered}}$.

Figure 5.3 provides a sample of frequent sets extracted from the data introduced in Figure 2.10 on page 15. The first number in the row gives a unique reference of a frequent set and the last number its frequency. In Figure 5.3, the last two patterns (Frequent sets 14 and 15), which contain

```
Input: Log $\mathbf{r}$, frequency, coverage and perfectness thresholds $\gamma, \kappa, \pi_r$
Output: Summary $FF$ and uncovered entries $\mathtt{r}_{\text{uncovered}}$ of log $\mathbf{r}$,
1. Find the collection $CFS$ of closed frequent sets w.r.t. the frequency
   threshold $\gamma$
2. Select summary $FF = \{S \mid S \in CFS \wedge \mathcal{C}_{\text{filt}}(S, \mathbf{r})\}$ w.r.t. the coverage
   threshold $\kappa$ and the perfectness threshold $\pi_r$
3. For all $S \in FF$ do
4.     For all $T \in FF \setminus \{S\}$ do
5.         if $T \subset S$ then
6.             $FF = FF \setminus \{S\}$
7.         fi
8. $\mathtt{r}_{\text{uncovered}} = \{e \in \mathbf{r} \mid \nexists S \in FF$ such that $S \subseteq e\}$
9. output $FF$, $\mathtt{r}_{\text{uncovered}}$
```

Figure 5.2: An algorithm for Comprehensive Log Compression.

```
...
5 {Destination:123.12.123.12, SPort:xx, Service:a_daemon, Src:B1}   10283
6 {Destination:123.12.123.12, Proto:tcp, Service:a_daemon, Src:B1}    10283
7 {Destination:123.12.123.12, Proto:tcp, SPort:xx, Src:B1}  10283
8 {Destination:123.12.123.12, Proto:tcp, SPort:xx, Service:a_daemon}    10283
9 {Destination:123.12.123.13, SPort:xx, Service:a_daemon, Src:B1}   878
10 {Destination:123.12.123.13, Proto:tcp, Service:a_daemon, Src:B1}    878
11 {Destination:123.12.123.13, Proto:tcp, SPort:xx, Src:B1}  878
12 {Destination:123.12.123.13, Proto:tcp, SPort:xx, Service:a_daemon}    878
13 {Proto:tcp, SPort:xx, Service:a_daemon, Src:B1}   11161

14 {Destination:123.12.123.12, Proto:tcp, SPort:xx, Service:a_daemon, Src:B1} 10283
15 {Destination:123.12.123.13, Proto:tcp, SPort:xx, Service:a_daemon, Src:B1}   878
...
```

Figure 5.3: A sample of frequent sets extracted from a firewall log.

five attributes each, have five subpatterns (Frequent sets 5 – 8 and 13, and Frequent sets 9 – 13). Figure 5.4 gives the corresponding closed sets. Let us assume that they all satisfy the constraint $\mathcal{C}_{\text{filt}}$ and they are initially included to the summary. However, closed sets 14 and 15 are specialisations of set 13 and thus linked to it according to inclusion lattice and removed from the summary. Only the set 13 is finally a filtering pattern.

There are several algorithms [55] that can be used for finding closed sets (Figure 5.2, Line 1). We have adopted a solution that first finds so-called *frequent free sets* and then produces their closures [13, 14]. This is efficient since the freeness property is anti-monotonic, i.e., a key property for an efficient processing of the search space. Also, if compared to Apriori-like

```
13 {Proto:tcp, SPort:xx, Service:a_daemon, Src:B1}   11161
14 {Destination:123.12.123.12, Proto:tcp, SPort:xx, Service:a_daemon, Src:B1} 10283
15 {Destination:123.12.123.13, Proto:tcp, SPort:xx, Service:a_daemon, Src:B1}   878
```

Figure 5.4: A sample of closed sets extracted from a firewall log.

Table 5.1: Three selected frequent sets found from a firewall log.

No	**Destination**	**Proto**	**SPort**	**Service**	**Src**	**Frequency**
1.	*	tcp	xx	a_daemon	B1	11161
2.	255.255.255.255	udp	xx	1234	*	1437
3.	123.12.123.12	udp	xx	B-dgm	*	1607

algorithms [4], with this algorithm it is possible to minimise the number of data base scans [126, 12].

Furthermore, other condensed representations have been proposed [25] including the δ-free sets [33], the $\vee$-free sets or the Non Derivable Itemsets [14, 23, 24]. They could be used in even more difficult contexts (very dense and highly-correlated data). Notice however, that from the end user's point of view, these representations do not have the intuitive semantics of the closed sets.

5.3 CLC usage scenarios

An example of the summary created by the CLC method from a firewall log can be seen in Table 5.1. It shows three patterns with high frequency values in the firewall log introduced in Figure 2.10 on page 15. The union of the supports of these patterns covers 91% of the data in the log. The fields marked with '*' do not have a value in the pattern and match all values in the data. For example, in Table 5.1 the field 'Destination' of Pattern 1 gets two different values on entries matched by it. These values are included in the specialisations of the corresponding closed set of Pattern 1 (Figure 5.4, Closed sets 14 and 15). The supports of the specialisations fully cover the support of Pattern 1.

In Table 5.2 the filtering pattern, Pattern 1, has been expanded by showing its specialisations as patterns 1.1 and 1.2. In a graphical browser, the patterns could also be drawn as nodes with appropriate connections.

A filtering pattern or its specialisation can be linked to its support set in original log **r**. The link can be implemented, for example, as a query, which fetches the support of the pattern. For example, a query for Pattern 2 of Table 5.1, would fetch entries that include field-value pairs

Table 5.2: The most covering pattern expanded to show its subpatterns.

No	Destination	Proto	SPort	Service	Src	Frequency
1.	*	tcp	xx	a_daemon	B1	11161
— 1.1.	123.12.123.12	tcp	xx	a_daemon	B1	10283
— 1.2.	123.12.123.13	tcp	xx	a_daemon	B1	878
2.	255.255.255.255	udp	xx	1234	*	1437
3.	123.12.123.12	udp	xx	B-dgm	*	1607

Destination:255.255.255.255, *Proto:udp*, *SPort:xx*, and *Service:1234*. The answer contains 1437 entries, which could be analysed further. For example, an expert might want to know the values in field *Src* and their frequencies in the answer and in the whole database.

An objective of the CLC method is to select the most informative patterns as starting points for navigating the log and its summary. What the most informative patterns are depends on the application and the task at hand. The filtering patterns are informative in the sense that they represent large amounts of entries and can be used to provide information about value distributions of varying fields in these entries. For example, a port scan might be captured by a closed set containing quite a lot of constant information, e.g., the source address in the *Source* field and label *"rejected"* in the *status* field. The entries in the support set of the port scan closed set contain varying values in field *DPort*, i.e., the destination port. These values cover the numbers of ports in the destination computer.

On the other hand, entries that are left to the compressed log, are possible signs of single anomalous or intrusive events and may be interesting as such. For example, if the values of field *DPort* in the entries in the support of the closed set related to a port scan do not contain all the port numbers, and if in the compressed log there is an entry that contains the same source address, the label *accepted* and one of the missing port numbers in field *DPort*, it might be an indication that the scan found an open port. In such a case, an expert should verify whether there is any more traffic between the port scan source and the server where the open port was found.

The summary may contain several patterns, whose supports overlap. Some of the supports of such patterns may be even totally covered by the others, and the redundant patterns could be removed without affecting the compression. The set of filtering patterns is not further optimised with respect to its size or coverage; all the filtering patterns found are included in the summary for browsing. The optimisation would be possible but tedious due to constraints that are applied in computation of frequent sets.

Minimum coverage and perfectness seem to be useful constraints to find good and informative filtering patterns for log summaries. The supports of selected patterns must be large to achieve good filtering effectiveness and the selected patterns must be perfect to be understandable for an expert and to lose as little information as possible, when the pattern is applied in filtering. If the perfectness of the pattern is 100%, it does not lose any information.

Selection of the most informative sets can also be based on the optimality with respect to coverage. It is possible that an expert wishes to see only the n most covering patterns or most covering patterns that together cover more than $m\%$ of the data. Examples of optimality constraints are considered in [139, 140].

5.4 Experiments

5.4.1 Experiment objectives

The purpose of these experiments is to evaluate the performance of the CLC method. The evaluation addresses the following questions:

- Is it possible to obtain useful results with the CLC method?
- How robust is the CLC method — or is it sensitive to parameter values or their combinations?
- How do changes in individual parameters affect the performance?
- How good are the results compared to some simpler alternative?
- How fast is the algorithm and how do different factors affect running times?
- How usable is the algorithm from the end user's point of view?

5.4.2 Method evaluation criteria

To evaluate CLC compression power experimentally, I define indicators to describe the size of the output. The output consists of a summary — i.e., the set of filtering patterns — and entries that are not covered by any of the filtering patterns. The indicators count patterns and entries in the output and make those relative by comparing them to the size of the original log. The *relative result size* allows comparison of results between data sets and throughout the experiment series.

Definition 5.7 (size of summary) The *size of summary* FF is $|FF|$, the number of filtering patterns in FF. □

Definition 5.8 (support of summary) The combined *support of summary* FF of log $\mathbf{r}$ is defined as $\text{SSupport}(FF, \mathbf{r}) = \{e \,|\, e \in \text{supp}(S, \mathbf{r})$ for any $S \in FF\}$. □

Definition 5.9 (uncovered entries) The set of *uncovered entries* $\mathbf{r}_{\text{uncovered}}$ of log $\mathbf{r}$ with summary FF is defined as $\mathbf{r}_{\text{uncovered}} = \{e \in \mathbf{r} \mid e \notin \text{SSupport}(FF, \mathbf{r})\} = \mathbf{r} \setminus \text{SSupport}(FF, r)$. The *number of uncovered entries* is $|\mathbf{r}_{\text{uncovered}}|$. □

Definition 5.10 (result size)) Given a log $\mathbf{r}$ and summary FF produced by CLC for $\mathbf{r}$, the *result size* is $\text{size}(FF, \mathbf{r}) = |\mathbf{r}_{\text{uncovered}}| + |FF|$. The *relative result size* is $\text{rsize}(FF, \mathbf{r}) = \text{size}(FF, \mathbf{r})/|\mathbf{r}|$. □

In the compression phase of CLC, the algorithm removes all the entries that are in the support of some of the filtering patterns in the summary. If the largest of such filtering patterns does not include some $field : value$ items of a removed entry, then information related to such items is lost. This is measured with *information loss*. A definition of *maximal cover* of an entry is used to formally define information loss.

Definition 5.11 (maximal cover) The *maximal cover* of an entry e by summary FF is defined as $\text{maxCover}(e, FF) = max(|S| \,\big|\, S \in FF \wedge S \subseteq e)$. □

Definition 5.12 (information loss) The *information loss* of CLC, with respect to given log $\mathbf{r}$ and its summary FF, is defined as $\text{iloss}(FF, \mathbf{r}) = (\sum_{e \in \text{SSupport}(FF, \mathbf{r})}(|e| - \text{maxCover}(e, FF)))/\sum_{e\in\mathbf{r}} |e|$. □

5.4.3 Method parameters

The CLC method requires three parameters: thresholds for frequency, coverage and perfectness. A pattern is included in the summary only if it fills the corresponding three minimal constraints. Possible value ranges for the thresholds are

- Frequency threshold $\gamma \in \{1, 2, \ldots, |\mathbf{r}|\}$,
- Coverage threshold $\kappa \in \{1, 2, \ldots, \sum_{e\in\mathbf{r}} |e|\}$, and
- Relative perfectness threshold $\pi_r \in [0, 1]$.

```
...
56773;6May2000;  0:00:15;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC2;123.123.123.123;udp;;;;;;;;;;
56774;6May2000;  0:00:38;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC18;123.123.123.123;udp;;;;;;;;;;
56777;6May2000;  0:01:27;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC19;123.123.123.123;udp;;;;;;;;;;
56779;6May2000;  0:02:18;eth-s2p1c0;fw.xyz.com;log;accept;ser2;ABC2;123.123.123.123;udp;;;;;;;;;;
56780;6May2000;  0:02:40;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC2;123.123.123.123;udp;;;;;;;;;;
56781;6May2000;  0:02:59;eth-s2p1c0;fw.xyz.com;log;accept;serABC;ABC10;255.255.255.255;udp;;;;;;;;;;
56782;6May2000;  0:03:01;eth-s2p1c0;fw.xyz.com;log;accept;serABC;ABC18;255.255.255.255;udp;;;;;;;;;;
56783;6May2000;  0:03:21;eth-s2p1c0;fw.xyz.com;log;accept;abc_daemon;ABC10;321.321.321.321;tcp;;;;;;;;;;
56784;6May2000;  0:03:21;eth-s2p1c0;fw.xyz.com;log;accept;abc_daemon;ABC10;321.321.321.321;tcp;;;;;;;;;;
56786;6May2000;  0:04:26;eth-s2p1c0;fw.xyz.com;log;accept;ser3;ABC2;123.123.123.123;udp;;;;;;;;;;
56787;6May2000;  0:04:31;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC2;123.123.123.123;udp;;;;;;;;;;
56790;6May2000;  0:05:35;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC12;123.123.123.123;udp;;;;;;;;;;
56794;6May2000;  0:07:43;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC10;123.123.123.123;udp;;;;;;;;;;
56795;6May2000;  0:08:19;eth-s2p1c0;fw.xyz.com;log;accept;ser3;ABC18;123.123.123.123;udp;;;;;;;;;;
56796;6May2000;  0:08:28;eth-s2p1c0;fw.xyz.com;log;accept;ser1;ABC12;123.123.123.123;udp;;;;;;;;;;
56798;6May2000;  0:08:59;eth-s2p1c0;fw.xyz.com;log;accept;serABC;ABC10;255.255.255.255;udp;;;;;;;;;;
56799;6May2000;  0:09:01;eth-s2p1c0;fw.xyz.com;log;accept;serABC;ABC18;255.255.255.255;udp;;;;;;;;;;
...
```

Figure 5.5: An example of test data.

If the number of fields, i.e., cardinality of entries, is constant throughout a log, then the coverage threshold $\kappa \in \{1, 2, \ldots, |\mathbf{r}| \cdot |e|\}$ and the absolut perfectness threshold $\pi \in \{0, 1, \ldots, |e|\}$. In the test data the number of fields changes over time between daily logs but it is constant in each log file. In the following experiments perfectness threshold is given as a minimum size of a filtering pattern π or as a maximum difference from the size of the entry π_d. For example, in a case where the entry size is 21 and at least 19 $field : value$ pairs are required for filtering patterns, the perfectness threshold can be given in one of the forms $\pi = 19$ (absolute), $\pi_r = 90\%$ (relative) or $\pi_d = 2$ (difference).

5.4.4 Datasets

The performance of CLC was experimentally evaluated with firewall log sets. Firewall log data was chosen as a test data for many reasons: There is plenty of data, the data is recorded for human experts, the data contains deterministic value combinations and the data needs to be stored for and analysed after a longer time period.

Firewalls have been protecting all business domains for a decade and personal firewalls are also spreading to home computers. In an enterprise, firewalls also separate network segments which are used for different purposes like factory network, research networks and financial networks. Each firewall produces thousands or even millions of log entries per day. This data needs to stored, for example, for after-the-fact analysis purposes, if a security incident takes place.

Figure 5.5 gives an anonymised sample slice of the data. Each data entry contains fields like entry number, date, time, interface, origin of the

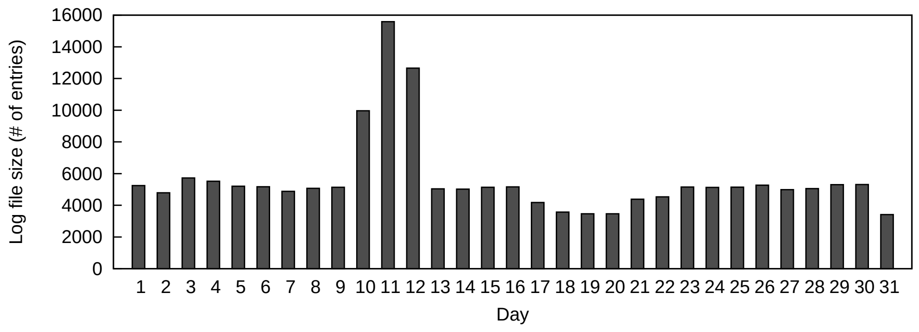


Figure 5.6: Daily log entry frequencies of data from small firewall.

entry, type, firewall action, attended service, packet source, packet destination, used protocol, and so on. The logs were dumped from a Firewall-1 proprietary database to ASCII files where the fields were separated by semicolons. As can be seen in Figure 5.5, there can be several fields that are empty. Different firewall rules and actions log different fields.

Each firewall log entry contains several values, which can be deterministically connected via firewall and system parameters. For example, in Figure 5.5, there is a functional dependency of attended service and protocol used. However, this does not apply in general, because the service might use several protocols, depending on its operation. Moreover, these dependencies change over time since system parameter values and firewall rules are changed.

Many of the attacks against information systems, especially the most interesting and dangerous ones, are unique. They take advantage of previously unknown vulnerabilities in the system. Therefore firewalls are often configured to store all the possible data in detail.

I evaluated the methods developed in this work with two data sets from two separate firewalls.

The first one, called *small firewall data* or *small data*, was recorded by a firewall protecting a small research network used for testing network applications. The data contains all entries collected during one month. The data has been divided into files, one file per day. Figure 5.6 depicts the number of entries per day. As can be seen, most of the time the number of entries is in the range from 5000 to 6000. Exceptional activity has occured during days 10, 11 and 12. The maximum frequency has been achieved on day 11 and is 15, 587. There have also been some low-activity days where the frequency has been under 4000 entries.

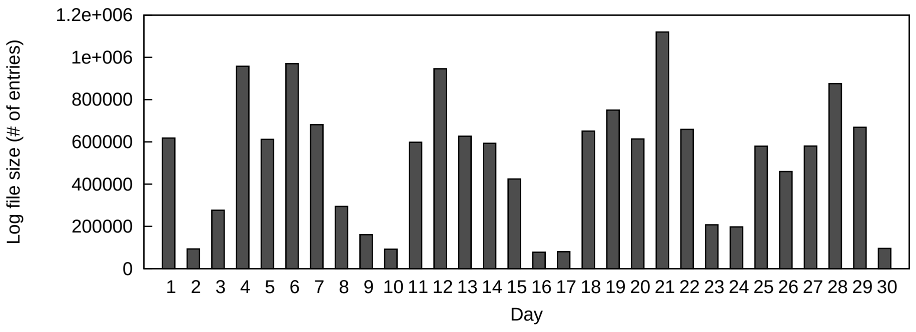


Figure 5.7: Daily log entry frequencies of data from large firewall.

The other data set, called *large firewall data* or *large data*, was generated by a set of firewalls guarding an enterprise network. If compared to the small firewall data, the volume and heterogeneity of traffic was much larger. This can be seen in Figure 5.7. The number of entries per day ranges from 77,674 to 1,119,850. The activity has a strong weekly cycle. It is lower on weekends (days 2, 3, 9, 10 and so on) and higher throughout working days.

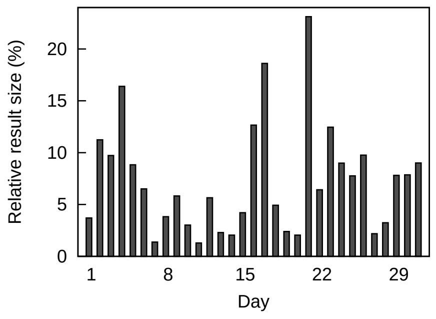
In the large data set, the entries have similar fields to the small data set (Figure 5.5). However, each field has a much larger variety of values, and in general there are no systematically empty fields. Here, the volume and heterogeneity of the data challenge the CLC method and therefore the large data set is used as a stress test for the method.

5.4.5 Example of performance

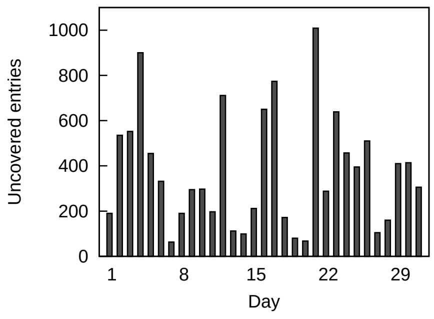
The experiments begin by ensuring that the method works at least in some cases. First the CLC method is applied with a reasonable parameter combination to both data sets and results are evaluated.

For the small data set the following threshold values were selected:

- Frequency threshold $\gamma = 100$ — If a pattern occurs four times an hour on an average, it is so frequent that it is included into a summary.
- Coverage threshold $\kappa = 5000$ — If the pattern has 20 items it has to occur in 250 entries to be included into a summary. Correspondingly for a pattern of 10 items occurences in 500 entries and a pattern of 5 items occurences in 1000 entries are required.
- Perfectness threshold $\pi = 19$ — In small firewall data the threshold value lets values in two fields to be left open in filtering patterns.



(a)



(b)

Figure 5.8: Relative result sizes (a) and the number of uncovered entries (b) of the small data set compressed with $\gamma = 100, \kappa = 5000$ and $\pi = 19$.

In all of these experiments, I have left out fields for the entry number and entry time. By definition they are changing so that they will not be included in the frequent sets. Leaving them out makes the algorithm execution more efficient.

The relative result sizes of daily data in small data set are depicted in Figure 5.8(a). The average relative result size of daily data files is 7%, the lowest is 1% and the largest 23%. The method compresses days $10, 11$ and 12 quite effectively also in terms of uncoverd entries (Figure 5.8(a)). These days contain the peak in the number of log entries. The log entry burst is caused by a misconfigured or broken process. It causes the firewall to record more than 500 almost identical entries in an hour. After CLC identifies and removes those entries, the rest of the data is quite similar to all the other days.

All the summaries except one contain just 3 or 4 filtering patterns (not shown). The only exception is the summary of day 31, which contains only one filtering pattern. The number of uncovered entries is depicted in Figure 5.8(b) Interestingly, when the entries covered by the summaries have been removed, the weekly activity cycle can be seen. The weekend days (6 and 7, 13 and 14 and so on) have fewer uncovered entries left in the data. A clear exception to this is Sunday the 21^{st}, for which the relative result size was also the worst. This is due to an efficiently distributed port scan, in which not only the scanned port is changing, but it is implemented by several tens of IP addresses in different parts of internet address space. CLC does not summarise it with perfectness threshold $\pi = 19$ since there are more than two varying fields in these entries. However, it is clearly

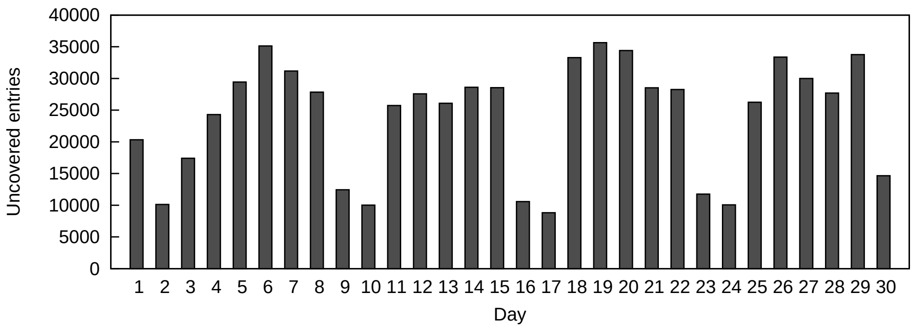


Figure 5.9: Number of uncovered entries of the large data set compressed with $\gamma = 100, \kappa = 5000$ and $\pi_d = 2$.

visible in the result for a human observer. Similar kinds of attacks can also be found in logs of days $4, 12$ and 17, each of which has more than 700 uncovered entries.

Corresponding experiments were carried out also with the large data set. The relative result sizes of the large data set (not shown) comply with those of the small data set. The average relative result size of the large data is 7% and the range is $[3\%, 15\%]$. Again the most effective compression is achieved on those days $(4, 6, 12, 21$ and $28)$ that contain a peak in the number of log entries. The reason for the peaks is partly the same as in the small data, i.e., a misconfigured or broken process. However, in each peak there are also signatures of prevented attacks.

As with the small data, the uncovered entries reveal a weekly cycle in the logging activity, which reflects the user activity in the monitored network (Figure 5.9). The CLC results of the large firewall data with these parameters contain more than $10,000$ uncovered entries. However, the set of uncovered entries still contains regularities that could be summarised with patterns by setting the thresholds lower. The effects of parameter values will be evaluated shortly.

5.4.6 Robustness of the CLC method

For practical applications it is important that the method always returns some results, even with an arbitrary parameter value combination. How CLC copes with this requirement is studied by executing CLC with parameter combinations that cover the possible value ranges.

The possible parameter ranges were defined in Section 5.4.3. In the case of the small data set, the selected parameter values were

- Frequency threshold $\gamma \in \{1, 10, 50, 100, 1000\}$,
- Coverage threshold $\kappa \in \{1, 100, 500, 1000, 5000, 10000, 20000, 50000\}$, and
- Perfectness threshold $\pi \in \{1, 15, 17, 19, 20, 21\}$.

The method was executed with all the parameter value combinations on each file. There were 7812 test runs altogether. In daily files there were $9 - 11$ entry fields which had constant values and $8 - 10$ fields which had several values. The *time* and *entryId* fields were left outside the analysis.

All the test runs returned with some results. At worst the method could not find any filtering patterns or it found few very short ones, which removed all the entries but represented only a small part of the information that the entries contained. Figure 5.10 shows the relative result sizes (Definition 5.10) for all the 7812 runs as a function of the three parameters.

The relative result size as a function of the frequency threshold is depicted in Figure 5.10(a). The average relative result size is around 20% with low frequency values. From there, the average increases and almost doubles at 1000. The number of filtering patterns drops since the relative frequencies then are very high. A common data file size in the small data is around 5500 entries. In a set of that size, if an item frequency is more than 100 it occurs in 2% of the entries and if it is 1000, in 20% of the entries. With such a threshold there are only short frequent patterns, which — in most test cases — are pruned from the summary by the perfectness threshold.

The median of the relative result size of all experiments grouped by the frequency threshold is significantly lower than the average. Since the data points contain results of all the possible threshold value combinations, the results of unreasonable parameter value combinations are included and degenerate the average.

As was said earlier, the coverage and frequency thresholds are closely related. With small coverage thresholds the average relative result size is around 10% (Figure 5.10(b)). When the threshold is increased to 5000, the relative result size begins to climb up fast. Coverage threshold value 5000 corresponds to frequency threshold value 263 with perfectness threshold 19. Test cases executed with large coverage threshold values are the main reason for the large difference between median and average of results grouped by the frequency or the perfectness threshold.

With regard to perfectness, the best compression is achieved when the threshold is low (Figure 5.10(c)). This is because the algorithm is then

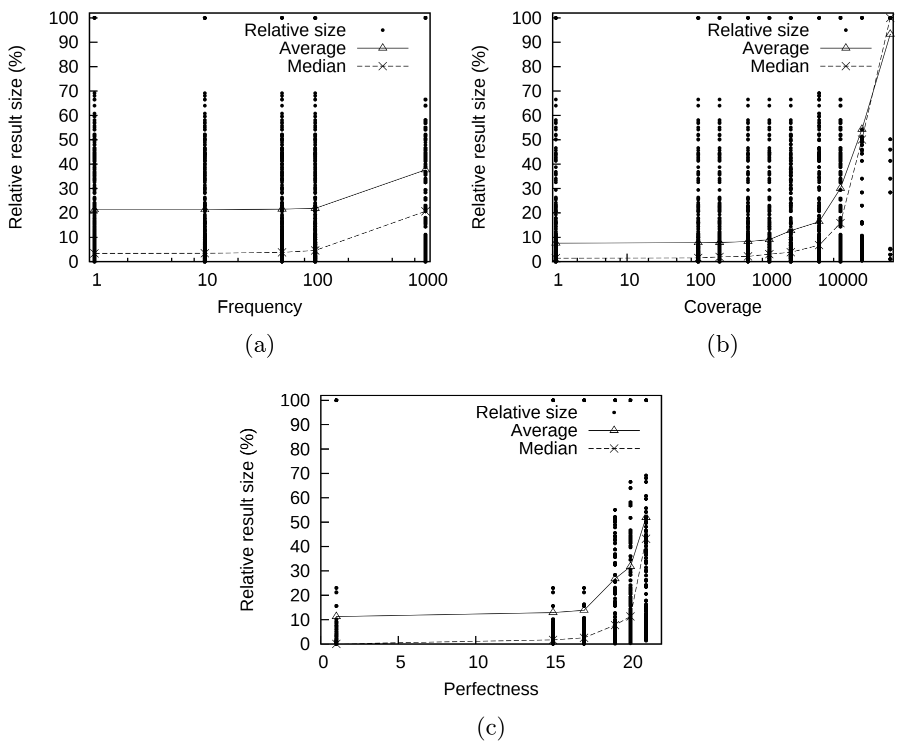


Figure 5.10: Relative result sizes of all the test runs of the small data set as a function of the frequency (a), coverage (b) and perfectness (c) thresholds.

allowed to select short filtering patterns to the summary. This kind of pattern easily represents a large part of the data. For example, a set of three simple filtering patterns each including only one item $\{protocol : udp\}, \{protocol : tcp\}, \{protocol : html\}$, would filter away most of the entries from almost any firewall log. Such a summary is not informative. A human observer can not get any idea of what has happened in the firewall.

The large data set was then analysed with parameter value combinations which covered the ranges of frequency and perfectness thresholds. Based on the results with the small firewall data (Figures 5.10(a) and 5.10(b)) the effect of coverage threshold correlates with the effect of the frequency threshold. Therefore, the coverage threshold was left out from the tests by letting its value be constantly $\kappa = 1$.

- Frequency threshold $\gamma \in \{50, 100, 500, 1000, 5000, 10000, 50000, 100000\}$,

- Coverage threshold $\kappa = 1$, and
- Perfectness threshold $\pi_d \in \{6, 4, 2, 0\}$.

Because the entry size varies from day to day in the large data set, it is more useful to give the perfectness threshold as the distance from the full entry size.

The frequency threshold values start from 50. This is quite low compared to the size of the data files. It varies in relative range [0.004%, 0.064%]. (Correspondingly, the absolute frequency threshold 1 of small data was in relative range [0.006%, 0.029%].) Smaller thresholds would have represented only small fragments of the data and they would probably have required quite long execution times. The algorithm searching for frequent closed sets is inherently sensitive to low frequency thresholds which often exponentially increase the amount of frequent sets found.

The method was executed with all the parameter value combinations for each file. In total there were 992 test runs. The *time* and *entryId* fields were again left out from the analysis. In daily files there was only one entry field which had a constant value.

As in the case of the small data set, all the test runs returned with some results. The results were well in line with the small data set (Figure 5.11). When the frequency threshold was high, the method lost much of its compression power that it had with small frequency threshold values. On the other hand, with frequency threshold $\gamma \leq 500$ and perfectness threshold $\pi_d \geq 2$, the average relative result size is less than 10.5%.

When the perfectness requirement was loosened, the method produced better compression results. An interesting step in improvement of relative result size can be seen when the perfectness is released from $\pi_d = 0$ to $\pi_d = 2$. The average relative result size drops even with large frequency threshold values from 100% to 40%. A reason for this is that the method identifies in this data set a small pattern set representing filtered snmp traffic. The frequency of these entries is more than one entry in a second and they represent a significant fraction of entries.

As a conclusion, the method appeared to be robust with respect to the data sets used. It was nearly always able to provide some kind of summary and reduction of data. At worst, when the frequency or coverage thresholds were high, the method could not find filtering patterns and did not achieve any compression. The execution times were decent and the laptop used had no difficulties of any kind to handle the processes. Only when the frequency threshold was set low, the execution time increased remarkably. Execution times and memory usage are discussed in more detail in Section 5.4.9.

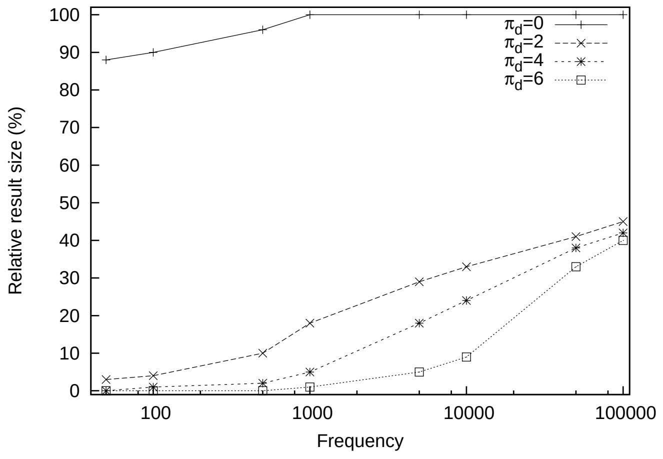


Figure 5.11: Average relative result sizes of the large data set as functions of the frequency threshold with different perfectness threshold values (π_d) given as difference from the full entry.

5.4.7 Effects of parameters

The results generated in the robustness tests were studied in more detail to find out how changes in parameter values affect the quality of results. In robustness testing the method was executed with all parameter value combinations. For evaluation of effects of parameter values, an analysis perspective was changed: the analysis was done for each threshold at a time. The values of the other two thresholds were fixed to some reasonable value combination and the analysed threshold was allowed to change in the range of possible values. This corresponds more closely to the real world use cases of the method. The relative result size and information loss (Definition 5.12) are used to analyse the results. Recall that information loss is defined as the relative amount of discarded items when compared to the original log.

The frequency threshold clearly affects the result size (Figure 5.12(a)). When the frequency threshold grows, the result becomes larger, since higher thresholds prune out more potential filtering patterns. The effect on the information loss is slighter: the information loss decreases with larger frequency thresholds (Figure 5.12(b)). The explanation for both observations is that with larger frequency threshold values there are fewer filtering patterns and more uncovered entries included in the result.

A growing coverage threshold also increases the result size (Fig-

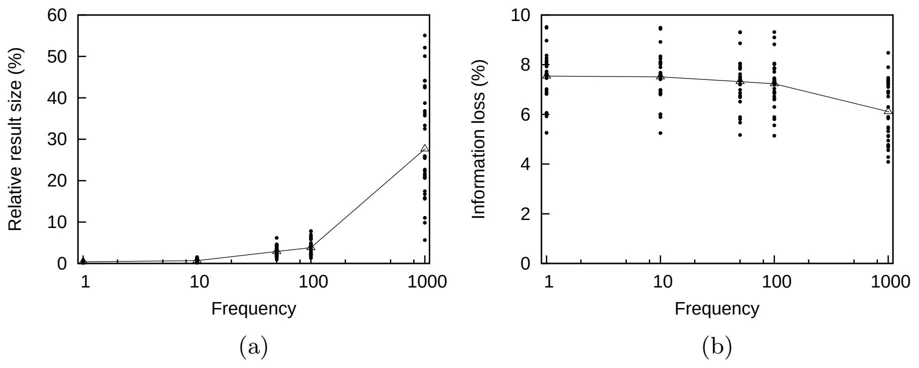


Figure 5.12: Relative result size (a) and information loss (b) of the small data set as functions of the frequency threshold. The line connects averages of each threshold. Coverage and perfectness thresholds were set to $\kappa = 1$ and $\pi = 19$.

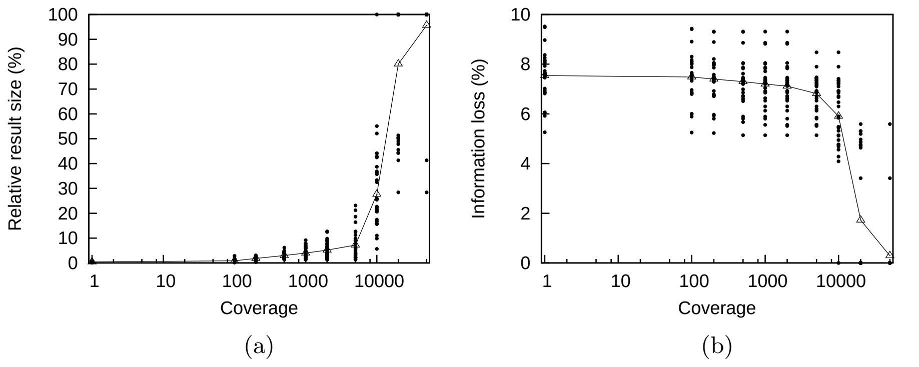


Figure 5.13: Relative result size (a) and information loss (b) of the small data set as functions of the coverage threshold. The line connects averages of each threshold. Frequency and perfectness thresholds were set to $\gamma = 1$ and $\pi = 19$.

ure 5.13(a)). As was mentioned earlier, it correlates with the effects of the frequency threshold. The information loss has an inverse shape (Figure 5.13(b)). When the coverage threshold is low, result sizes are small and information loss is at its highest. When the coverage threshold gets high the information loss decreases towards zero. As with the frequency threshold, this happenes since fewer filtering patterns and more uncovered

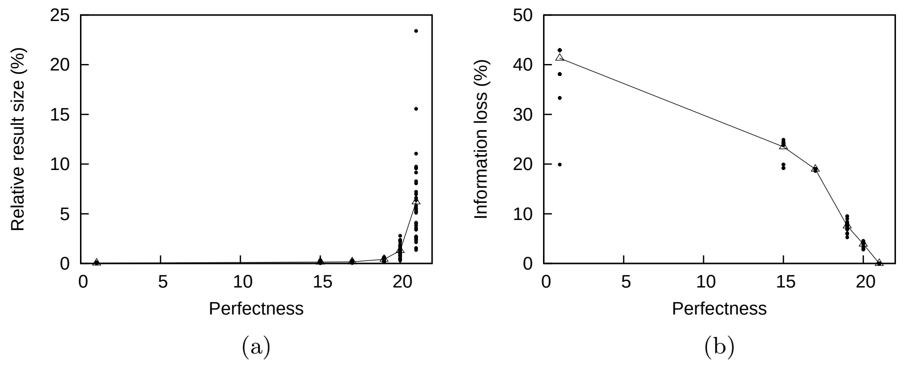


Figure 5.14: Relative result size (a) and information loss (b) of the small data set as a function of the perfectness threshold. The line connects averages of each threshold. Frequency and coverage thresholds were set to $\gamma = 1$ and $\kappa = 1$.

entries are included in the result.

The perfectness threshold strongly affects the information loss. When it is low, the algorithm produces small results with short filtering patterns (Figure 5.14(a)). It also loses a lot of information (Figure 5.14(b)). When the perfectness increases, less information is lost and results become larger. This happens since a higher perfectness threshold enforces longer filtering patterns. Thus fewer entries are removed and a larger portion of removed entries is represented in patterns that pruned them.

The results with the large data set were well in line with those of the small data. The similarity is evident for results obtained with identical frequency threshold values (Figure 5.15). The only larger difference is in relative result size with the frequency threshold $\gamma = 1000$. It appears to be caused by the size of the small data set files. There were several days where no patterns were found with that threshold, increasing the average result size.

When the whole frequency threshold range of the large data is considered, the result size continues to increase and information loss to decrease when the frequency threshold increases. The intersection point of relative result size and information loss is at the frequency $\gamma \approx 240$ (Figure 5.15). It is a good candidate for a frequency threshold, as a compromise between the contradicting objectives of achieving small results with small information loss.

Information loss with different perfectness threshold values obtained

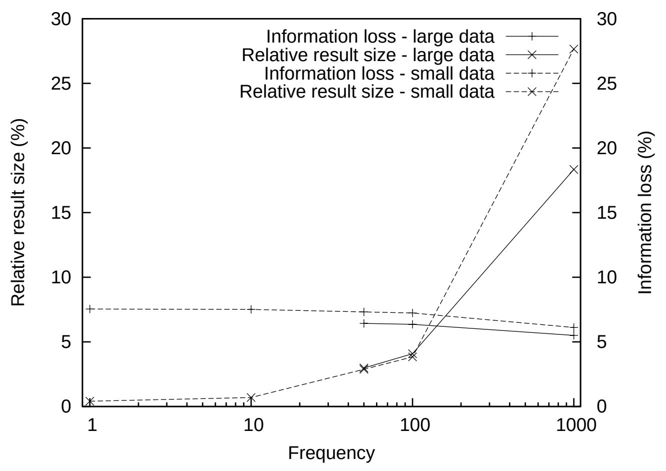


Figure 5.15: Average relative result size and information loss of the large and small data set as a function of the frequency threshold with perfectness threshold $\pi_d = 2$.

from the large data set is shown in Figure 5.16 as a function of frequency threshold. When the perfectness threshold π_d is relaxed — the difference from the full entry size gets larger — the compression is done with shorter filtering patterns. Thus the information loss increases. Simultaneously the relative result size decreases (see Figure 5.11 on page 72). The ultimate results are achieved when the perfectness $\pi_d = 0$, when no information is lost. These results correspond to those obtained from the small data.

5.4.8 CLC compared to a simple reduction

To study the general quality of the results, a simple baseline compression was defined and the CLC results were compared to it. A *reduct* of each daily log was used as a baseline.

The reduct of a log is the collection of all unique entries after the removal of *time* and *entryId* fields. In the worst case it contains all the entries in the data and at its best all entries are identical. Using the notation and definitions of this chapter, the reduct can be defined as follows.

Definition 5.13 (reduct) A summary R is a *reduct* if $\gamma = 1 \wedge \pi_d = 0$. □

In general, relative reduct sizes were quite small with the small data set, i.e., from 1% to 23%. From all the small data set results produced in

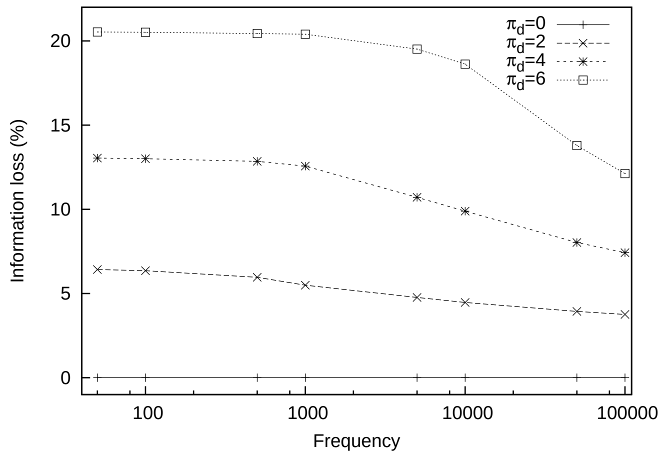


Figure 5.16: Average information loss of the large data set as functions of the frequency threshold with different perfectness threshold values (π_d) given as difference from the full entry.

the robustness test, 55% were smaller than the reduct of the log. When the parameter ranges were set to the values that seemed to produce the best results, i.e.,

- Frequency threshold $\gamma = 10$, and
- Perfectness threshold $\pi = 19$

then all the result sizes were below the reduct size, the largest being 79% of the reduct and 3% of the original log.

The relative reduct sizes of the large data set were higher: from 45% to 94%. From all the robustness test results of the large data set, 71% of results were smaller than the corresponding reduct. Again, all the best results with

- Frequency threshold $\gamma = 50$, and
- Perfectness threshold $\pi_d = 2$,

were smaller than the corresponding reducts. Now the largest result size was only 10% of the corresponding reduct size and 6% of the log size.

In both cases the information loss was small. With logs in the small data set, information loss was in the range [5.3%, 9.4%] and with logs in the large data set in the range [5.7%, 7.0%].

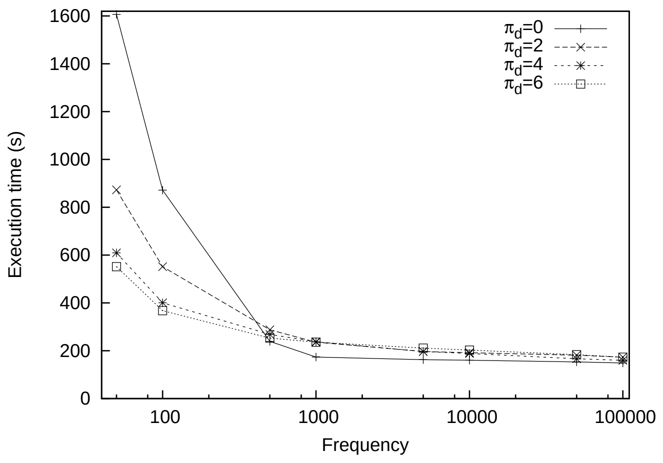


Figure 5.17: Average execution time of the large data set as functions of the frequency threshold.

The effects of the coverage threshold correlate with those of the frequency threshold. Therefore, in both cases the coverage threshold was set to so small a value that it did not affect the computation.

5.4.9 Efficiency

In practical applications the algorithm execution time and memory consumption play an important role. To clarify the time and memory consumption, I will study the execution of robustness tests in more detail.

All the tests were run on an IBM Thinkpad T43 laptop with an Intel Pentium M 1.86 GHz processor provided with 2GB of main memory and running Linux operating system v2.6.18.

The CLC execution time of the small data set was less than two seconds with thresholds $10 \leq \gamma$. This is quite feasible for any on-line application. The execution times were longer only with frequency threshold $\gamma = 1$, when the algorithm extracted all possible patterns. Thus the average was around four seconds and at the worst the extraction took twenty seconds. Changes in coverage and perfectness thresholds had in practice no effect at all on the overall execution times.

The same applies to the large data. The frequency thresholds have an effect on the execution time when other parameters have only a minor effect on it (Figure 5.17). Again a small frequency threshold introduces longer execution times. The perfectness threshold has a major effect only

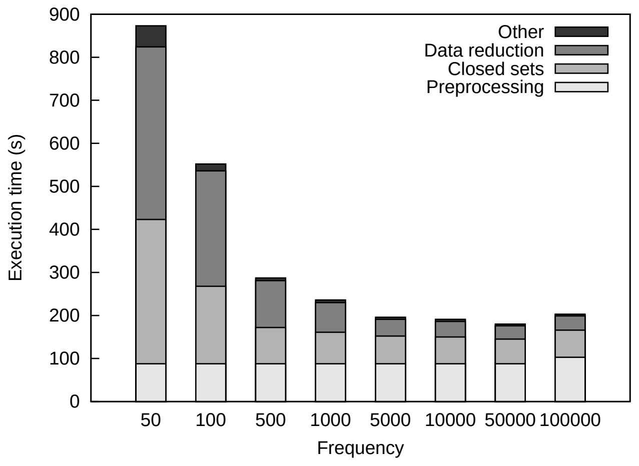


Figure 5.18: Execution time of the large data divided to algorithm phases with different frequency threshold values.

with small frequency threshold values, where the number of long patterns is larger.

The absolute times of test runs with the large data — on the average from a couple of minutes to twenty minutes — are feasible for batch processing but not for online applications.

When we analyse where the time is spent inside the algorithm, a reason for the importance of frequency threshold becomes evident (Figure 5.18). Most of the time is spent in pre-processing the data and in search for closed sets in it. In preprocessing the implementation prepares the data for the algorithm searching for closed sets: removes fields left out from the analysis and fields with constant values, compiles $field : value$ pairs to unique integers, and so on. With lower frequency thresholds almost as much time is spent in data reduction — i.e. removing all the entries in the union of filtering pattern supports. When plenty of closed sets are found, the set of possible filtering patterns becomes large, and matching patterns against each of the entries takes longer time.

To reduce the time spent in data reduction the implementation could be optimised. Currently it uses a sequental search to find a matching filtering pattern for each log entry. An optimised solution could be built, for example, on a trie structure of items in filtering patterns [41, 91] or some sort of optimised automata [5].

The correspondence between execution time and number of filtering

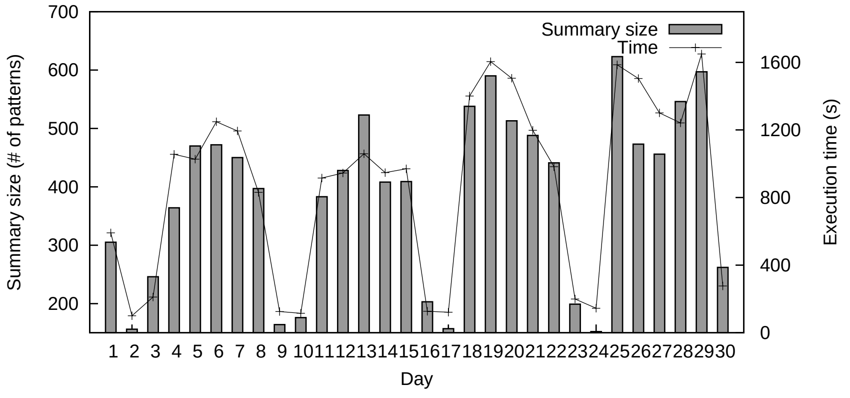


Figure 5.19: Correspondence between CLC execution time and summary size of the large data set.

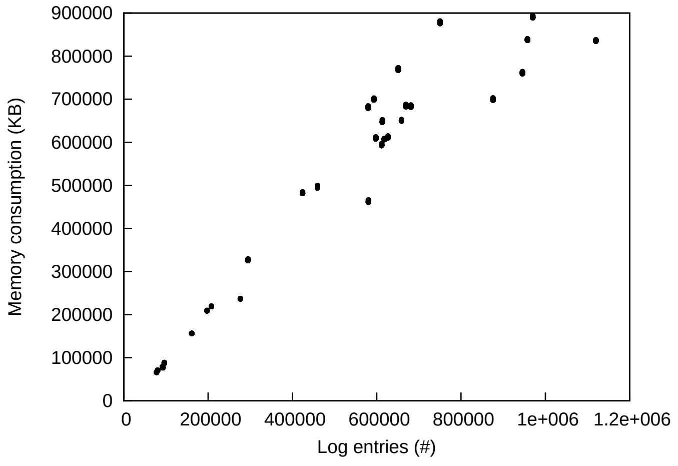


Figure 5.20: Memory consumption as a function of the number of input entries of the large data set.

patterns found is clearly visible (Figure 5.19). The CLC algorithm was executed with parameters $\gamma = 50$, $\kappa = 1$ and $\pi_d = 2$. The execution time varies with the summary size.

The memory consumption of this implementation of the CLC algorithm correlates almost linearly with the input size (Figure 5.20). A rough rule of thumb states that with the large data each entry consumes about 1KB of main memory.

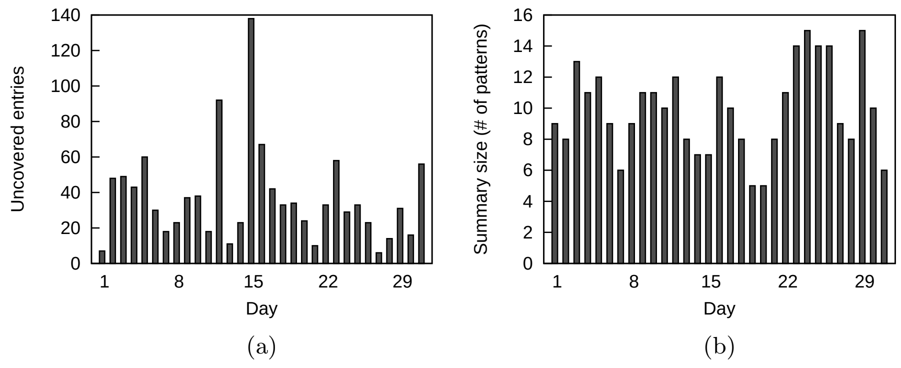


Figure 5.21: The uncovered entries (a) and the summary size (b) of the best results of the small data set, achieved with $\gamma = 10, \kappa = 100$ and $\pi = 19$.

5.4.10 Usability

The experiments are concluded by studying the best results produced in robustness tests. The selection is based on a trade-off between compression power, information loss and summary size.

The best results for the small data set in terms of compression power and information loss are achieved with parameters $\gamma = 10, \kappa = 100$ and $\pi = 19$. There the number of uncovered entries is below 100 in all except one case (Figure 5.21(a)), which a human observer can easily analyse. The average information loss was 7.5%, a minimum 5.3% and a maximum 9.4%.

The third aspect of the usability of the results is the size of the summary. If it is very large, it just turns the problem of analysing a large log to a problem of analysing a large summary. In case of the best small data results, the summaries have at most 15 patterns (Figure 5.21(b)). This is an understandable representation for an expert.

Considering the large dataset, a set of $1,000,000$ entries is too large for almost any log viewer tool to present and handle. When reduced to $4000 - 20,000$ entries, the tools can offer assistance in the analysis. This compression for the large data set is achieved with the CLC method with parameters $\gamma = 50, \kappa = 1$ and $\pi_d = 2$ (Figure 5.22(a)).

The information loss was practically constant at 6% With such a loss, an expert can see phenomena behind removed entries from the filtering patterns.

The summary sizes vary from day to day reflecting the weekly activity cycle (Figure 5.22(b)). The achieved summaries for week days, where the

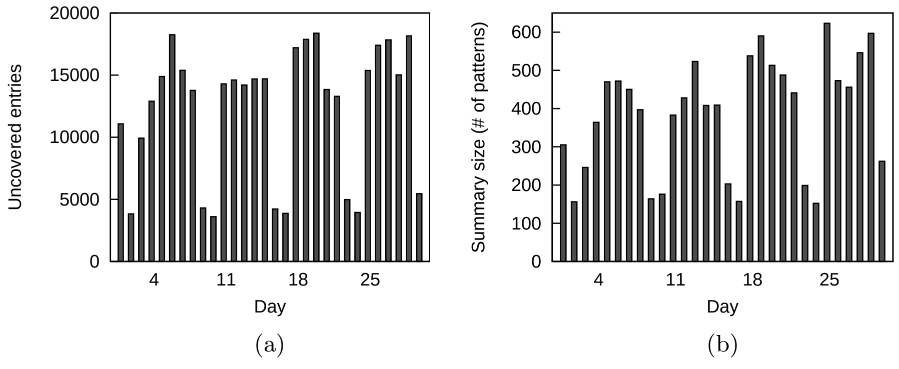


Figure 5.22: The uncovered entries (a) and the summary size (b) of the best results of the large data set, achieved with $\gamma = 50, \kappa = 1$ and $\pi_d = 2$.

size varies in range $300 - 630$ are quite large to analyse at a glance. With proper browser, however, analysis is feasible.

The filtering pattern sets in summaries for different days are overlapping. The re-occurring patterns found can be used as rules in, e.g., entry correlation, where uninteresting entries are filtered away. They can also be used in firewall rule design to decide whether or not a rule creates a log entry when it fires. New and anomalous patterns can also be identified by comparing the summaries to each other. These are probably the most interesting patterns, since they might represent some new attack or network disturbance to be dealt with.

5.5 Applicability and related work

The experiments show the usefulness of the CLC method. Its idea is simple, its computation efficient and its filtering power remarkable. With a small number of filtering patterns in a summary, it can represent the majority of the data of a large log file. CLC representation gives a human expert or a computationally more intensive algorithm a chance to continue with the data that does not contain too common and trivial entries.

The scalability of the CLC method is good. Closed sets can be extracted from highly correlated and dense data, i.e., in contexts where the computation of the whole collection of frequent patterns is intractable [126, 12, 164, 128]. This enables the use of CLC in analysis of log files, where many field values are highly correlated and frequent sets are long containing most of the items in entries.

A comparison of frequent and closed sets derived from some days of small data set shows how much smaller a collection of closed sets is with regard to a corresponding collection of frequent sets [55]. In the CLC method the collection of closed sets is further diminished in number when the perfectness and coverage thresholds are used in selecting the most informative and understandable sets to summary.

A collection of maximal frequent sets [48, 9, 111] that contains all most specific frequent sets derived from the data can also be considered to be used in CLC. The collection of maximal frequent sets is a subset of the collection of closed sets [21]. As the maximal frequent sets can be computed without extracting all the frequent sets or even all the closed sets, they can be computed from the data that contains several tens of items in the largest frequent sets [21].

A collection of closed sets contains also sets that are not maximal. In some cases these non-maximal closed sets are more informative. For example, in Figure 5.4 the two sets of length five are maximal and the first set, whose specialisations they are, is not. When a collection of closed sets and the CLC algorithm are used, the generalisation is selected to the summary (see Table 5.1). If a collection of maximal frequent sets were used, the summary would have contained the two maximal sets instead.

In data sets that contain several tens or even hundreds of items in the largest frequent sets, the maximal sets may be a good choise for CLC. Especially, if the items do not correlate and the closeness property does not decrease the search space efficiently. In what conditions and with what kind of data maximal frequent sets would challenge closed sets in CLC are interesting questions that are left open for further studies.

CLC supports on-line analysis of large log files. This is often required for a system state identication task that is common in many telecommunications management processes on the tactical level where signs of different types of incidents are searched for. The idea of summarising frequently repeating information is quite effective in an environment where lots of log contents are signs of normal operation and thus repeated from day to day. Normal operation is not interesting but the deviations from it are.

The CLC type of representation is general with respect to different log types. Closed sets can be generated from most of the logs that have structure and contain repeating symbolic values in their fields; like, for example, in Web Usage Mining applications [93, 145]. The main difference between the CLC method and those applications is the objective of the mining task. Most of the web usage applications try to identify and somehow validate common access patterns in web sites. These patterns are then used to, e.g., optimise the site. The CLC method, however, does not say anything about semantic correctness or relations between the frequent patterns found. It only summarises the most frequent value combinations in entries.

Some related approaches have been published after the introduction of the CLC method [55]. Most of them aim at approximating a collection of frequent item sets with a smaller subset. Boundary cover sets [2], δ-clusters and their representative patterns [162], and set of K-representatives [163] all summarise a set of frequent patterns by grouping patterns to homogeneous sets and selecting a representative pattern for each set. The distance measure and algorithm used in grouping and selecting the representative pattern varies from one method to another. The difference between these approaches and the CLC method is that instead of reducing analysed patterns, the CLC method aims at reducing the amount of log entries, i.e., transactions that a human expert needs to study.

Collections of frequent itemsets have also been used to characterise clusters produced by a probabilistic clustering using mixtures of Bernoulli models [72, 73]. In these works, data has been first clustered using probabilistic clustering. A collection of frequent sets has been computed for each data cluster from the data points included in the cluster. For each frequent set collection, a set of maximal frequent sets is selected and used to represent and describe the cluster. The cluster specific frequent set collections differ from each other and characterise the data from which they were computed [72]. The extracted maximal frequent sets summarise the marginal distributions in the clusters compactly and describe the clusters in domain specific terminology, for example, in the naming scheme for chromosomal regions used in literature [73].

Two related approaches summarise dense categorical transaction and log data bases. The SUMMARY algorithm [159, 160] searches for each of the transactions for the longest frequent itemset that is included in the transaction. The set of these so called *summary itemsets* is called a *summary set* of the data base and presented as its summary. A summary set may not be unique because a transaction may support more than one summary itemsets. However, the algorithm can be enhanced to find also the alternative summary sets.

An approach for log data summarisation [26, 27] searches for an informative set of frequent sets that covers a set of selected suspicious log entries. The approach turns the summarisation problem to a dual-optimisation where a summary is characterised with *information loss* and *compaction gain*. The summarisation can be done either by clustering log entries or using a greedy algorithm to find a suboptimal set of informative frequent sets with respect to introduced measurements.

The log data summarisation method differs from the CLC also in the search direction. The CLC computes the set of filtering patterns and removes those log entries that are in support of at least one filtering pattern. In the log data summarisation method of [26, 27], the method starts from the data entries by attaching to each of them the most representing frequent set and then optimising the set of these representatives. With large log files this might become quite tedious.

Although CLC shares characteristics with these two approaches — SUMMARY and log data summarisation — their objective is different. These approaches aim at presenting a summary of a whole selected set of data entries while CLC aims at finding sets of similar, common entries that can be removed from the data since the frequency and other properties of the entry sets are more interesting than the entries themselves.

Similarly as related approaches, CLC is a lossy compression method. It loses information about entry times and order as well as values in fields not covered by filtering patterns. However, if it is possible to store the original data and the filtering patterns in the CLC summary can be linked to it, it is possible to analyse those details, as well.

In CLC the set of selected patterns may contain many overlapping patterns. Depending on the task and the information need, the interesting patterns may differ. It is possible that the most frequent patterns filter away information that would reveal an anomaly in less frequent field values. For example, if there are fewer servers than provided services in the network, the most frequent patterns may contain values pointing to different servers. If a suspicion arises that a certain service is maliciously misused, the user interface should also provide the possibility to view less frequent selected patterns including pointers to services.

The use of the CLC method requires only setting of default values for three thresholds when it is installed. It does not require any other kind of *a priori* knowledge. The experiments also show that the CLC method is not sensitive for non-optimal parameter values. From the experiments one may suspect that, due to dependency between frequency and coverage thresholds, two thresholds might be sufficient in many cases. Probably importance of the coverage threshold would be higher, if the amount of fields in entries would be larger.

The features that a user interface provides for browsing the summary and the data are very important. However, they are left outside the scope of this work.

CLC versus requirements The CLC method appears to fulfill requirements set for the data mining and knowledge discovery methods and tools summarised in Section 4.5 quite well. The method does not require data-mining-specific knowledge when it is used (*Easy-to-use methods, Easy to learn*). On run time when the method is applied, the only statistical figure that is necessary to understand is the frequency of a pattern. When the method is installed, default values have to be set also to the coverage and perfectness thresholds. Otherwise the network expert operates on the concepts and log contents that are specific for his own domain (*Application domain terminology and semantics used in user interface*). The method efficiently provides immediate answers: summaries and a reduced set of analysed data (*Immediate, accurate and understandable results, Efficiency and appropriate execution time*). This speeds up the analysis task and reduces the iterations needed (*Reduced iterations per task, Increases efficiency of domain experts*). The CLC method can be integrated into existing tools, for example, as a new view or a report (*Interfaces and integrability towards legacy tools*).

Only the requirements of *Adaptability to process information* and *Use of process information* are not directly addressed. However, the CLC method supports the domain expert in acquiring new information from the data. He can then compare this information to his existing knowledge about the system, its processes and their current state.

Chapter 6

Queryable lossless Log data Compression (QLC)

Large volumes of daily network log data enforce network operators to compress and archive the data to offline storages. Whenever an incident occurs — in system security monitoring, for example — that immediately requires detailed analysis of recent history data, the data has to be fetched from the archiving system. Typically the data also has to be decompressed before it can be analysed.

This kind of on-line decision-making tasks on the tactical level are challenging for data mining methods and the knowledge discovery process. There is not much time to iterate, data management requires a lot of effort even before the analysis can take place, and the network expert knows everything about, for example, network authentication server protocols and unix tools but only little about statistics and data mining. Critical resources available for data mining developers are missing (Figure 4.3).

As Chapter 5 already showed, it is possible to use closed sets to compress data. This chapter further elaborates the idea and modifies the presented methods to data archiving [56]. Closed sets can be used to create a representation that reduces the size of the stored log by coding the frequently occurring value combinations. The coding can be done without any prior knowledge about the entries. The compression does not lose any information and the original log file can easily be restored. The representation can also be queried without decompressing the whole log file first. This approach is more straightforward compared to the solutions proposed for semantic compression of databases [6] and documents [110].

6.1 Background and method overview

Log files that telecommunications networks produce are typically archived in compressed form. Compression is, in many cases, done with some general-purpose compression algorithm like the Lempel-Ziv compression algorithm (LZ) [165], Burrows-Wheeler Transform [22], or PPM (prediction by partial matching) [36] or with some algorithm designed specifically

```
*;11May2000;*;a_daemon;B1;12.12.123.12;tcp;; 4
*;11May2000;*;1234;*;255.255.255.255;udp;; 2
```

Figure 6.1: Two closed sets with frequencies derived from a firewall log excerpt (Figure 2.10). The fields marked with '*' do not have a value in the closed set. The last field of both sets contains an empty string.

for log data compression [135, 7, 143]. When the log files are restored and a query or a regular expression search for relevant entries is made, the whole archive must be de-compressed.

Another possibility to archive history logs is to insert them to a database management system first and then, after a certain period, compress the whole database table to a file that is inserted to a mass storage. Problems arise, when there is a need to analyse old backups. An expert has to find the correct media, de-compress the whole database and load it to the database management system. This can be problematic because the amount of data included in a database per day might be large. Thus its de-compression and uploading takes a lot of time.

Without compression the problem is to fit the database to mass storage media and still be able to manage the rapidly growing number of mass storage media. Selecting carefully what is stored can reduce the problem, but still there tends to be quite a lot of data written into archives. Selection might also lose important information, which is not acceptable, for example, in security application logs.

Figure 2.10 on page 15 shows an example excerpt from a database containing transactions that store firewall log entries produced by CheckPoint's Firewall-1. As can be seen, the transactions are filled with entries that share correlating value combinations but still there are some fields whose values are varying; e.g., there are *TIME* and *ID* fields that are changing from entry to entry.

Figure 6.1 lists closed sets that are used in data compression by QLC, the method to be proposed in this chapter. They contain those items that have the largest coverage, i.e., they cover the largest amount of field values in the table. These sets are output as *compression patterns* (Figure 6.2, first two lines). The method then goes through each entry. It compares the entry to the compression patterns. If the entry supports any of the patterns, values included in the pattern are removed from the entry and are replaced by a reference to the pattern. In Figure 6.2 the compressed data are shown by using an XML-tagged format. If the original table is needed, the formulae can be used to restore it completely — no information

```
<define p0> *;11May2000;*;a_daemon;B1;12.12.123.12;tcp;; </define>
<define p1> *;11May2000;*;1234;*;255.255.255.255;udp;; </define>
777; 0:00:23;<p0>
778; 0:00:31;<p0>
779; 0:00:32;B1;<p1>
780; 0:00:38;B2;<p1>
781; 0:00:43;<p0>
782; 0:00:51;<p0>
```

Figure 6.2: Compressed firewall log excerpt (Figure 2.10), with two patterns and six compressed entries.

will be lost.

6.2 Definitions and algorithms for QLC

6.2.1 Log compression

Here we define concepts that are needed in log compression and decompression phases.

Definition 6.1 (compression gain) The *compression gain* of a closed set S in a log $\mathbf{r}$ is defined by $\mathrm{cgain}(S, \mathbf{r}) = \mathrm{cov}(S, \mathbf{r}) - ((\mathrm{freq}(S, \mathbf{r}) \cdot n) + |S|)$ where n denotes the size of the reference to the pattern and $|.|$ denotes the cardinality of the closed set S. $\square$

Definition 6.2 (compression pattern) A closed set S is a *compression pattern* in log $\mathbf{r}$ with respect to frequency, coverage and perfectness thresholds γ, κ and π if it satisfies constraint $\mathcal{C}_{\mathrm{compr}}$, where $\mathcal{C}_{\mathrm{compr}}(S, \mathbf{r}) \equiv \mathcal{C}_{\mathrm{minfreq}}(S, \mathbf{r}) \wedge \mathcal{C}_{\mathrm{mincov}}(S, \mathbf{r}) \wedge \mathcal{C}_{\mathrm{minperf}}(S, \mathbf{r}) \wedge \mathrm{cgain}(S, \mathbf{r}) > 0$. $\square$

Definition 6.3 (compressed log) A *compressed log* $\mathtt{r}_{\mathrm{compr}}$ derived from log $\mathbf{r}$ with respect to frequency, coverage and perfectness thresholds γ, κ and π consists of a set of compression patterns, i.e., *compression formulae* $\mathtt{r}_{\mathrm{compr}}.\mathit{CF} = \{S \mid \mathcal{C}_{\mathrm{compr}}(S, \mathbf{r})\}$ and a set of *compressed entries* $\mathtt{r}_{\mathrm{compr}}.\mathit{entries} = \{(e \setminus p) \cup \{\mathit{ref}(p)\} \mid e \in \mathbf{r} \wedge p \in \mathtt{r}_{\mathrm{compr}}.\mathit{CF} \wedge e \in \mathrm{supp}(p)) \wedge (\forall q \in \mathtt{r}_{\mathrm{compr}}.\mathit{CF} \text{ s.t. } e \in \mathrm{supp}(q) : |p| \geq |q|)\} \cup \{e \in \mathbf{r} \mid \nexists p \in \mathtt{r}_{\mathrm{compr}}.\mathit{CF} \text{ s.t. } e \in \mathrm{supp}(p)\}$. $\square$

The compressed log is not uniquely defined if there are several longest compression patterns applicable to any entry.

Log compression is conceptually a straightforward operation (Figure 6.3). After the frequent closed sets in the given log have been identified, the algorithm selects those whose compression gain is positive. The

Input: Log **r**, frequency, coverage and perfectness thresholds γ, κ and π
Output: Compressed log $\mathtt{r}_{\text{compr}}$

```
1. Find the set of frequent closed sets $CFS$ from $\mathbf{r}$
2. Select compression formulae
   $CF = \{S \mid S \in CFS \wedge \mathcal{C}_{\text{compr}}(S, \mathbf{r})\}$.
3. Output $CF$
4. Compress log ($CF$, $\mathbf{r}$) and output the result        // Figure 6.4
```

Figure 6.3: An algorithm for log compression.

Title: Compress log
Input: Compression formulae *CF*, log **r**
Output: Compressed entries $\mathtt{r}_{\text{compr}}.entries$ of log **r**

```
1. for each entry $e \in \mathbf{r}$ do
2.     $CF_e = \{p \mid p \in CF \wedge e \in \text{supp}(p)\}$
3.     if $CF_e \neq \emptyset$ then
4.         $p_{compr} = p$ s.t. $(p \in CF_e) \wedge (\forall q \in CF_e) : |p| \geq |q|$
5.         $e' = (e \setminus p_{compr}) \cup \{ref(p_{compr})\}$
6.         Output $e'$
7.     else
8.         Output $e$
9. od
```

Figure 6.4: An algorithm for compressing log entries.

compression gain is evaluated with respect to the original log **r** (cf. Definition 6.1). The order in which patterns are selected thus does not affect the result. On the other hand, this process obviously can result in a set of patterns where some patterns are redundant and could be removed with no actual effect on the compression.

The compression formulae are then used to remove recurrent value combinations away from the log entries. The removed values are replaced with a reference to the pattern that was used to identify them. The replacement is the most specific, i.e., the longest compression pattern that applies to the log entry is used as is shown in Figure 6.4. The compression pattern might be ambiquous if there are more than one pattern that are as long as the longest pattern. Also then, the algorithm selects only one as a compression pattern.

The compression ratio of the algorithm is not optimal. This is because it is not able to handle compression patterns that are partially overlapping. The algorithm selects the most specific one, and makes the compression of an entry with it. It would also be possible to use many patterns per entry. In some cases this could lead to improved compression ratio and minimise the size of compression formulae. However, the search for the best pattern combination per entry would be more complex.

A simple modification that reduces the size of compression formulae is to compress the log entries first, mark all the compression patterns that were used and remove all those patterns that were not used. This would require switching the order in which the entries and the compression patterns are output (Figure 6.3, Lines 3 and 4).

A structure of a compressed file can also be optimised further in a domain like firewall logs where each entry contains a unique identifier and entries are stored in an increasing order of the identifiers. A simple optimisation could be that all the entries compressed with the same pattern are grouped together. Thus it is enough to mark the entry groups and include a reference to the compression pattern only in the beginning of each group. The pattern references can be left out from the compressed entries. This would require changes in the Compress log algorithm (Figure 6.4, Lines 6 and 8 and at the end) where the output of compressed entries should be directed to the pattern specific buffers that are combined to output only after all the entries have been processed.

The computational complexity of the algorithm mainly depends on the algorithm that searches for closed sets and the algorithm that finds the most specific compression pattern for an entry. With this in mind we have done a series of experiments on factors that affect the execution of the algorithm. The results of these experiments are reported in Section 6.3.

The de-compression of log data is as simple as the compression. The algorithm goes through the compressed log entries and expands them with the values of the compression pattern that was used for compression. The entries may also be sorted if their order was changed due to structural optimisation.

6.2.2 Queries to compressed logs

One of the main advantages of the proposed compression method is its ability to support queries in compressed form.

Definition 6.4 (query) A *query* Q is a subset of `Items`. □

Definition 6.5 (query result) A *query result* RS of query Q on log $\mathbf{r}$ will contain all those entries $e \in \mathbf{r}$ that *match* the query: $RS = \{e \mid Q \subseteq e\}$. $\square$

While evaluating query Q, we want to extract all log entries $e \in \mathbf{r}$, such that $Q \subseteq e \iff Q = Q \cap e$. In compressed log $\mathbf{r}_{\text{compr}}$ all log entries $e \in \mathbf{r}_{\text{compr}}.\mathit{entries}$ are either stored as such or compressed by replacing some pattern $p \subseteq e$ with a reference $ref(p)$ and storing $(e \setminus p) \cup \{ref(p)\}$. Thus we do not have to test all the entries whether the query is included in them but we can take advantage of information whether the query is subset of a pattern used to compress an entry.

Given an entry $e \in \mathbf{r}$ denote the compression pattern applied to it by p. Thus $p \subseteq e$ and further $e = p \cup (e \setminus p)$. Thus $Q \subseteq e \iff Q = Q \cap e = Q \cap (p \cup (e \setminus p)) = (Q \cap p) \cup (Q \cap (e \setminus p))$. We have four options when evaluating query Q on entry e:

1. $Q = \emptyset$: Thus $Q \cap p = \emptyset$ and $Q \cap (e \setminus p) = \emptyset$ and an answer will be the whole log $\mathbf{r}$, since the $\emptyset \subseteq e$ for all entries $e \in \mathbf{r}$;
2. $Q \cap p = Q$: Thus all the entries $e \in \text{supp}(p, \mathbf{r})$ are included to the answer;
3. $Q \cap p \neq \emptyset$: Thus we need to evaluate the query against each of the entries $e \in \text{supp}(p, \mathbf{r})$ to see whether $(Q \cap p) \cup (Q \cap (e \setminus p)) = Q$;
4. $Q \cap p = \emptyset$: Thus we need to evaluate the query Q against each of the entries $e \in \text{supp}(p, \mathbf{r})$ to see whether $Q \cap (e \setminus p) = Q$

An algorithm for query evaluation is given in Figure 6.5. Query Q is matched against each of the compressed entries $e \in \mathbf{r}_{\text{compr}}.\mathit{entries}$. If the entry is not compressed with any pattern, then the query is matched to the entry, otherwise the algorithm needs to study both the compression pattern and the compressed entry.

If the entry contains a reference to a pattern, which is a superset of the query, then the entry is decompressed and included to the answer (Option 2, Line 7). If the query is overlapping with the pattern but not completely subset of it (Option 3, Line 9), the algorithm tests if the query is subset of the union of the pattern and the compressed query. Finally, if the intersection between the query and the pattern is empty, it is enough to test if the query is subset of the compressed entry (Option 4, Line 12).

Computational complexity of the algorithm is linear in the number of entries. However, the algorithm can be optimised in many ways. For example, the query can first be matched against each of the patterns and the result of the intersection (subset, disjoint, overlapping) can then be hashed

Input: Compressed log $\mathtt{r}_{\text{compr}}$, Query Q
Output: Query result $RS = \{e \in \mathbf{r} \mid Q \subseteq e\}$.

```
 1. $RS = \emptyset$
 2. **for** each entry $e' \in \mathtt{r}_{\text{compr}}.entries$ **do**
 3.     $e =$ NULL
 4.     **if** $\nexists$ pattern $p \in \mathtt{r}_{\text{compr}}.CF$ s.t. $e' \in \text{supp}(p)$ **then**
 5.         **if** $Q \subseteq e'$ **then**
 6.             $e = e'$
 7.     **elsif** pattern $p \in \mathtt{r}_{\text{compr}}.CF \wedge ref(p) \in e' \wedge Q \subseteq p$ **then**
 8.         $e = e' \setminus \{ref(p)\} \cup p$
 9.     **elsif** pattern $p \in \mathtt{r}_{\text{compr}}.CF \wedge ref(p) \in e' \wedge Q \cap p \neq \emptyset$ **then**
10.         **if** $Q \subseteq (p \cup e')$ **then**
11.             $e = e' \setminus \{ref(p)\} \cup p$
12.     **elsif** pattern $p \in \mathtt{r}_{\text{compr}}.CF \wedge ref(p) \in e' \wedge Q \cap p = \emptyset$ **then**
13.         **if** $Q \subseteq e'$ **then**
14.             $e = e' \setminus \{ref(p)\} \cup p$
15.     **if** $e \neq$ NULL **then**
16.         $RS = RS \cup \{e\}$
17. **return**($RS$).
```

Figure 6.5: An algorithm for query evaluation in a compressed log.

with the pattern reference as a key. Thus the algorithm does not need to repeat set intersection testing between the query and the pattern with every entry.

In a domain like firewall logs, where a field can have only one value in each entry, a query never matches to an entry that contains another value for the field included in the query. The algorithm can use this information for optimisation by identifying patterns containing $field : value$ items that disagree with the query. Entries that have been compressed with such a disagreeing pattern can be left out from the result without further inspection.

6.3 Experiments

6.3.1 Experiment objectives

The purpose of these experiments is to evaluate the performance of the QLC method. The evaluation addresses the following questions:

- Is it possible to obtain useful results with the QLC method?

- How do changes in individual parameters affect the performance?
- How good are the results compared to some simple alternative?
- How fast is the algorithm and how do different factors affect running times? How sensitive is the QLC method to data size?
- How efficiently can the results be accessed?

6.3.2 Experiment arrangements

The effectiveness of QLC can be measured by the result size and execution time. The QLC method does not lose any information, so the information loss is not measured here. In the following experiments, the result size has been measured by the resulting file size. Thus it is simple to compare the QLC results to the results of the well-known compression algorithm Lempel-Ziv [165] and its commonly used implementation named GNU zip or `gzip`.

When a compression algorithm is evaluated then typically both compression and decompression need to be studied. In the QLC case, the querying of compressed results is analysed. This is because the main usage for QLC is to archive data so that it can be accessed without decompression.

The analysed method parameters are the same as with the CLC: thresholds for frequency, coverage and perfectness. Their possible value ranges used here are also the same as with CLC (Section 5.4.3).

The QLC was evaluated using the same data sets as with CLC (Section 5.4.4). All the experiments were also made on the same laptop as the CLC experiments (Section 5.4.9).

6.3.3 QLC performance example

As an example of the QLC method performance it was applied with fixed parameter values to the small and large data sets.

The parameter values were selected based on experiences with the CLC. The intention was to ensure that the gain of each compression pattern was positive. With the small data set the thresholds were set as follows:

- Frequency threshold $\gamma = 10$,
- Coverage threshold $\kappa = 20$, and
- Perfectness threshold $\pi_d = 16$.

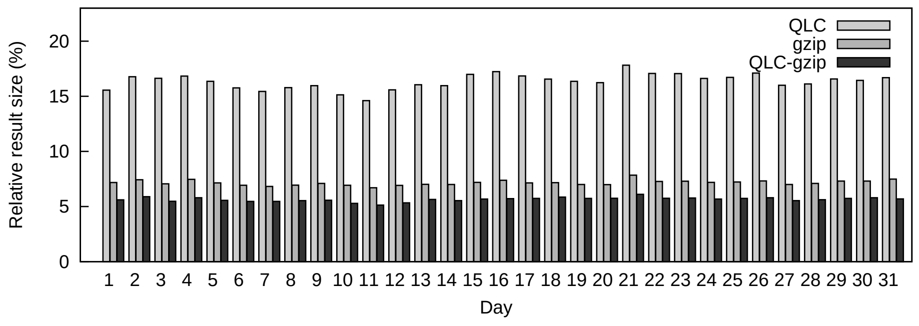


Figure 6.6: Relative sizes of QLC compressed and GNU zipped files and GNU zipped QLC results of a small data set ($\gamma = 10, \kappa = 20, \pi_d = 16$).

In contrast to the CLC experiments, here the perfectness threshold as a difference from the full entry is higher and does not dominate the analysis. This is because the compression formulae are not meant for a human observer, but for optimising the compression of the data. Thus the compression patterns do not need to be long and semantically meaningful.

First the QLC compression was performed with said parameters. The result size was 16% of the original file size on an average (Figure 6.6). As resulting files are in ASCII format it is possible to further compress them with `gzip`. The combined QLC-`gzip` compression result size was only 5% of the original size on an average — approximately one third of the QLC-compressed file size. Such small files are easy to handle and in most cases they fit into the main memory in the querying or decompression process. For the querying the most essential part of the QLC compressed file — the header with the compression patterns — is in the beginning of the compressed file and relatively short with respect to the original file. Decompression and query processing of the QLC-`gzip` compressed file is easy and efficient as will be seen later in Section 6.3.7.

The QLC compression results — the plain and the combined QLC-`gzip` compressed — of the small data set were compared to the compression results of *gzip, v. 1.3.5* (Figure 6.6). As can be seen, the combined compression result sizes are always the smallest. The average `gzip` compressed file size is 7% of the original log file size. The relative improvement from `gzip` to combined compression is 21% on an average.

The large data set was compressed with threshold values

- Frequency threshold $\gamma = 500$,

- Coverage threshold $\kappa = 1000$, and
- Perfectness threshold $\pi_d = 16$.

The results were in line with those of the small data set and thus not shown. The average result sizes were 26% of the original log file size for the QLC method, 8% for `gzip` and 5% for QLC-`gzip` combined. For example, with an original log file of 135.1MB (day 21), the QLC compressed result was 34.5MB, `gzip` result 8.6MB and combined result 5.9MB. The relative compression improvement from the `gzip` to combined results was on average 34%. This is a considerable reduction, for example, in archiving.

6.3.4 Effects of parameters

The meaningful value ranges for thresholds were the same as those for CLC evaluation (Section 5.4.6). The threshold values used for the large data evaluation were as follows.

- Frequency threshold $\gamma \in \{50, 100, 500, 1000, 5000, 10000, 50000, 100000\}$,
- Coverage threshold $\kappa \in \{100, 1000, 10000, 100000\}$, and
- Perfectness threshold $\pi_d \in \{16, 6, 2\}$.

All the log files of the large data set were compressed with all the threshold value combinations. Similarly as the CLC, the QLC method was able to produce some results with all the threshold value combinations.

The compression ratio was at its best, when the thresholds were at their lowest values. The effect of the frequency threshold is a good example of this (Figure 6.7). The largest frequency threshold value ($\gamma = 100,000$) is larger than some log file sizes. In these cases the algorithm did not find any compression patterns and the file was not compressed. These cases have been omitted from the results and there are not so many data points as with other frequency thresholds. Omission of the largest relative sizes affects the average so that it does not increase from the preceding value.

As with the CLC method, an effect of increasing the coverage threshold value corresponds to the effect of increasing the frequency threshold value (not shown). Increasing the perfectness threshold value beginning from $\pi_d = 2$, did not have any effect to the result sizes (not shown). This is probably characteristics for firewall data, where most of the variation in the most common log entries concentrates to one or two fields at a time:

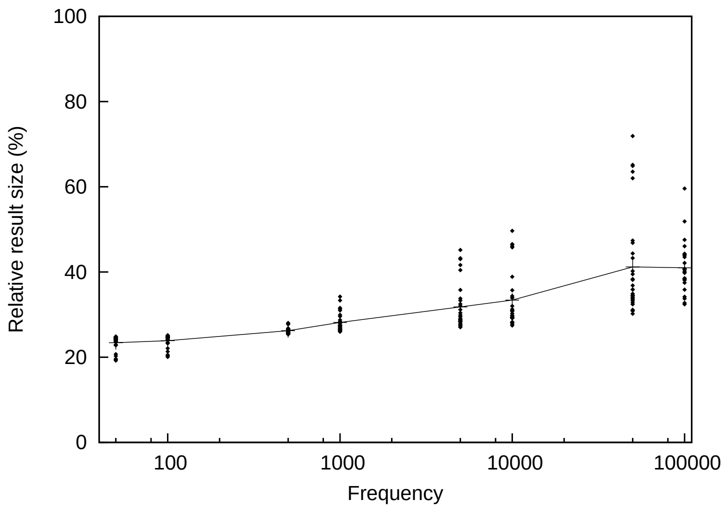


Figure 6.7: Relative sizes of QLC compressed results of the large data as a function of a frequency threshold. The coverage and perfectness thresholds are constants $\kappa = 100$ and $\pi = 16$.

for example, a port scan typically introduces entries where values in only one field vary.

The small data set was tested with the threshold values

- Frequency threshold $\gamma \in \{1, 2, 5, 10, 20, 50, 100, 200, 500, 1000\}$,
- Coverage threshold $\kappa \in \{1, 20, 50, 100, 500, 1000, 5000, 10000\}$, and
- Perfectness threshold $\pi_d \in \{16, 8, 6, 4, 2, 0\}$.

The QLC was applied to all the log files of the small data set with all the value combinations. The compression results of the small data set were well in line with the results of the large data set and are therefore not reported in detail.

6.3.5 QLC compared to `gzip`

Sizes of the QLC compression result files were compared to those of `gzip` and the sizes of the combined QLC-`gzip` compressed results, for all the parameter combinations above (Figures 6.8 and 6.9). The combined QLC-`gzip` compression always provides the smallest result files. Only with days 10, 17 and 30 of the large data set the size of `gzip` compressed file is inside

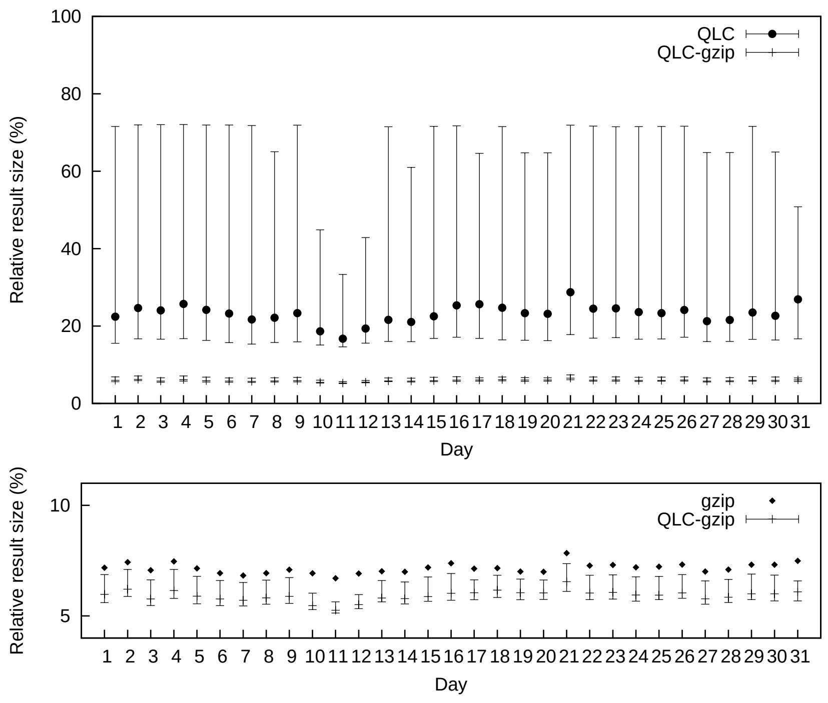


Figure 6.8: Average relative sizes of QLC and QLC-`gzip` compressed files (upper pane) and QLC-`gzip` and `gzip` compressed files (lower pane) of small data. 95% of data points are included inside error bars.

the value range that covers 95% of the QLC-`gzip` compression results of all the test runs in large data experiments.

In the large data set, the worst compression results are caused by the high threshold values (Figure 6.9). When the threshold values are too high, the QLC method does not find effective compression patterns, and outputs the input file as such. For days 10, 17 and 30, there are parameter combinations with which QLC was not able to compress the log file. In those cases in the combined compression, `gzip` was applied to the original file and the result was identical to that of plain `gzip` compression. In the experiments with small data there are no such combinations (Figure 6.8).

The results show that the QLC method is quite robust with respect to the parameter combinations. Especially combined QLC-`gzip` compression provides on average 26% better compression results than the plain `gzip`, calculated over all the large data experiments.

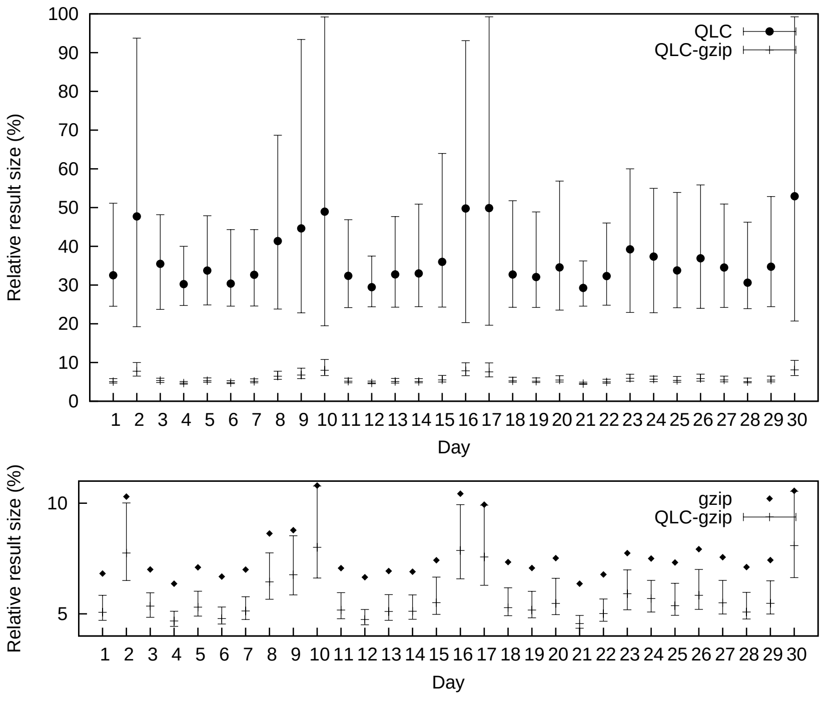


Figure 6.9: Average relative sizes of QLC and QLC-`gzip` compressed files (upper pane) and QLC-`gzip` and `gzip` compressed files (lower pane) of large data. 95% of data points are included inside error bars.

6.3.6 Compression efficiency

In earlier tests [56], the possible sensitivity of QLC to the size of the input data was studied. The QLC method was evaluated with logs of different sizes that included entries with a varying number of fields. The data used in these tests were a set of randomly selected daily logs from the small and the large data sets.

The results of these tests showed that the execution time of the QLC algorithm depends linearly on the number of fields and the number of log entries.

For this thesis, the efficiency of the QLC algorithm was further evaluated by applying it to the small and large data sets with all above mentioned parameter combinations. With the small data and a frequency threshold values $\gamma \geq 2$, the average execution time was always around one second. The largest maximum was 2.6 seconds with threshold $\gamma = 2$. Only with

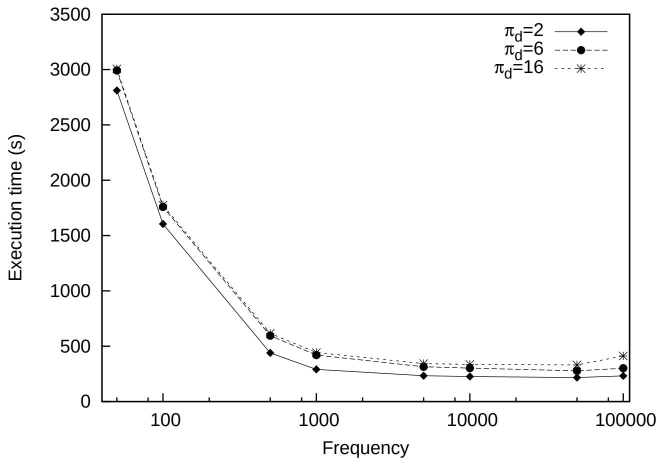


Figure 6.10: Average QLC compression times of large data as functions of frequency and perfectness.

the threshold value $\gamma = 1$, when all in data occurring item combinations were considered, the average increased to 3.5 seconds and maximum to 21.5 seconds.

The effect that the frequency threshold has on the execution times can be seen with the large data (Figure 6.10). The QLC implementation uses plenty of time with small frequency thresholds but the time consumption decreases rapidly with increasing threshold, and with a larger frequency threshold it does not change much. The time consumption correlates with the amount of closed sets. When there are more closed sets, the algorithm needs more time to search for them and more importantly, to do the data reduction, i.e., to find all instances of compression patterns (selected closed sets) in log entries.

Lowering the perfectness threshold π (i.e., increasing π_d, the difference from the full entry) also increases time consumption. This is also due to an increasing amount of compression patterns (Figure 6.10).

Data reduction is the most time-consuming part of the QLC algorithm (Figure 6.11). The time to search for instances of patterns in log entries seems to be almost exponential with respect to the number of selected closed sets. When the longest compression pattern is searched for, the patterns are matched to an entry in decreasing length order. When the algorithm finds a matching compression pattern, it removes the corresponding $field : value$ pair sets and replaces them with a reference to the pattern. This kind of

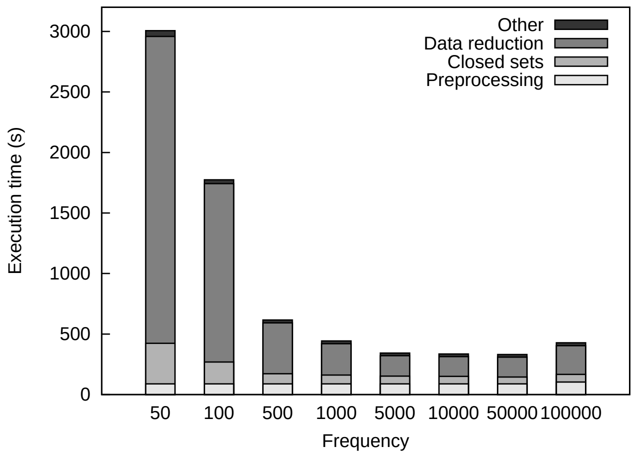


Figure 6.11: Average times of algorithm phases as a function of frequency.

sequential search for a matching pattern is inefficient. The matching fails for most entries several times before the correct pattern is found.

The slight rise in execution times with the largest frequency threshold (Figure 6.11) is explained by those logs, whose size was less than the largest threshold. The QLC method did not compress them since it could not extract any compression patterns. Their compression times with the smaller threshold values were much below the average.

The memory consumption (not shown) depends mostly on the number of log entries. The QLC implementation uses approximately 1KB of main memory per each log entry. This is inline with the observation made with CLC.

6.3.7 Query evaluation

Query test setups

The QLC compressed data files can be queried without a separate decompression operation. The efficiency of the query evaluation was tested by defining a set of queries in the form of *field:value* pairs and evaluating them on each of the QLC compressed data files. In evaluation an optimised implementation of the query algorithm given in Figure 6.5 was used.

The queries were defined so that they

- produced answers of different size,

Table 6.1: Query classes of large data set experiments. Detailed queries are given in Appendix B, Table B.2.

Id	Description	Queries in class	Queried items	Output size Min	Output size Median	Output size Max
A	Miscellaneous queries with small output	6 (1–6)	1	0	60	5092
B	Small queries with large and medium output	4 (7–10)	1–3	1512	7456	1119850
C	Small query with empty output	1 (11)	2	0	0	0
D	Set of increasingly specific small queries with regular expression	2 (12,13)	1–2	29	280	5336
E	Set of increasingly specific queries with large, almost constant output	5 (14–18)	1–6	20351	53469	128769
F	Set of increasingly specific queries with reducing output	4 (19–22)	1–5	0	2081	128769
G	Set of increasingly specific queries with constant output	3 (23–25)	1–3	295	890	32388
H	Set of increasingly specific overlapping queries with large output	5 (26–30)	1,3	11	331109	1038208
I	Set of increasingly specific queries with reducing output	4 (31–34)	3–6	0	15913	212003

- contained different amounts of queried *field:value* pairs, and
- contained both string literals and regular expressions in values.

For the large data, there were 34 defined queries, which can be grouped into nine classes (Table 6.1; detailed queries are given in Appendix B, Table B.2). Queries in classes *A, B* and *C* were not related to each other in the sense that the results were not overlapping. In the rest of the classes, the results of the queries are either partially overlapping or subsets of results of previous queries.

Query answers can be partially overlapping if the queries are partially overlapping. In class *H*, for example, queries *action:drop AND i/f_dir:inbound AND proto:udp* and *action:drop AND i/f_dir:inbound AND service:XYZ* may provide partially overlapping results.

If a query p is a superset of query q, then its answer is a subset of that of q. In classes *D-I* most of the queries are formed by adding an item to

Table 6.2: Query classes of small data set experiments. Detailed queries are given in Appendix B, Table B.1.

Id	Description	Queries in class	Queried items	Output size Min	Median	Max
a	Miscellaneous queries with a small output	5 (1–5)	1,2	0	0	878
b	One item query with the entire log file as an answer	1 (6)	1	3411	5132	15587
c	Set of partially overlapping increasingly specific queries with regular expression and small output	7 (7–13)	1–4	0	6	962
d	Set of increasingly specific queries with constant output	9 (14–22)	1–9	413	681	682
e	Set of increasingly specific queries with medium reducing output	8 (23–30)	1–8	0	1516	2806
f	Set of increasingly specific queries with medium reducing output	6 (31–36)	1–6	0	2338	4323

some of the shorter queries in the class. For example, the queries in class E are all specifications of the query *action:reject*, the second one being *action:reject AND i/f_dir:inbound*.

The size of an answer is a meaningful factor in query evaluation. Therefore, the queries have been defined to provide answers of different sizes. The median of the class shows the characteristic answer size in the class. In some classes, like classes F and I, the answer size reduces when the query is more specific.

Regular expressions can also be used in queries in place of field values. For example, in class D, we were interested in all the entries in which the user field was not empty. Such entries were found with a query *user:[a-zA-Z0-9]+*.

The corresponding query classification of the small data is given in Table 6.2 (detailed queries are given in Appendix B, Table B.1).

In the following experiments the threshold values for the QLC compression of large data were

- Frequency threshold $\gamma = 500$,
- Coverage threshold $\kappa = 50$ or $\kappa = 100$, and
- Perfectness threshold $\pi_d = 16$.

The coverage threshold $\kappa = 100$ was used in measuring effects of different frequency thresholds. In practice, those two threshold values are so close to each other that there is no difference between their effects.

The corresponding values for the small data were

- Frequency threshold $\gamma = 10$,
- Coverage threshold $\kappa = 20$, and
- Perfectness threshold $\pi = 5$.

The QLC queries were compared to queries applied to the log files compressed with `gzip`. For querying they were decompressed with `zcat`, forwarded to the process executing the query and the results were output to a file. The used search programs were `egrep` and a corresponding perl script.

A query $field_i : value_j$ was expressed to search programs as a regular expression. In the file format used, fields are identified by their position (column) in the file. To ensure that $value_j$ was matched for the correct field, the preceding fields were represented with regular expression '`[^;]*;`', which represents any string in a field and the field delimiter ';'. For example, a query with two items: *action:reject AND proto:icmp*, was expressed as a regular expression '`^[^;]*;[^;]*;[^;]*;[^;]*;[^;]*;reject;[^;]*;[^;]*;[^;]*;icmp`', which was then given for `egrep`.

Query test results

The first observation was that the size of the query answer is the factor that affects the evaluation time the most (Figure 6.12). The correlation between average query time and the number of found entries is almost one.

The second observation about the performance of the QLC was that the sorting of the entries to correspond with the original order in the log file was time consuming. The overhead caused by sorting was minimal with small answers and was more than 40% ot the total time with the largest answers (Figure 6.13).

The querying on QLC-`gzip` compressed files was a bit slower than with the QLC compressed files (Figure 6.13). The added overhead was introduced by the `zcat` program that decompressed the `gzip` compressed QLC file and forwarded it to the QLC query program.

Next, QLC was compared to the `egrep`-program that sequentially searches for given regular expressions in the data. As `egrep` is a compiled program and the QLC-query implementation is made with Perl — an interpreted script language — the setup could favor `egrep`. Therefore, I

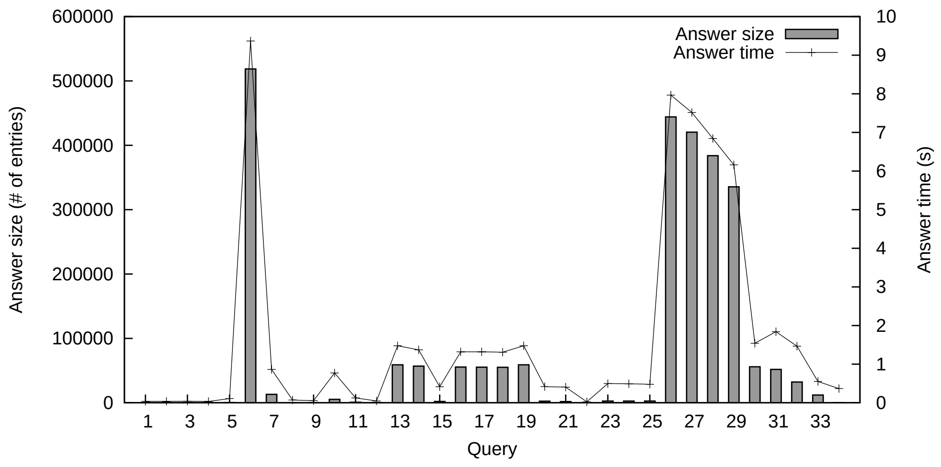


Figure 6.12: Average size of answers and QLC query times on large data.

also implemented a perl script *lGrep* that searches for given regular expressions in the data and included that in the comparison. Results for both are included in the figures since their behaviours turned out to be quite different.

The QLC query providing unsorted answers was nearly always the fastest option (Figure 6.14). When the answer was very large (queries 7 and 26 in the lower and query 31 in the upper pane of the figure), the QLC query was slower than `egrep` and *lGrep*.

The behaviour of `egrep` was quite strange with some queries. There is no single visible reason for its slow performance with queries $12, 24, 15, 28, 29$ and 30. The reason can possibly have something to do with the query evaluation, when the given value is further in the line (query 12) the amount of queried items is increasing and the answer is not diminishing (queries $27 - 30$) or something else (queries 15 and 24).

lGrep behaves quite steadily. The execution time of *lGrep* increases remarkably with seven queries $(12, 7, 26 - 30)$. Probably the regular expression searched for is more demanding in those queries. However, the execution time does not rise suddenly with any query as it does with `egrep`. Probably the regular expression evaluation has been optimised better in the perl interpreter than in `egrep`.

The results with a small data set were again well in line with those of large data (Figure 6.15). The QLC query with unsorted answers is the fastest, except when the answer is large.

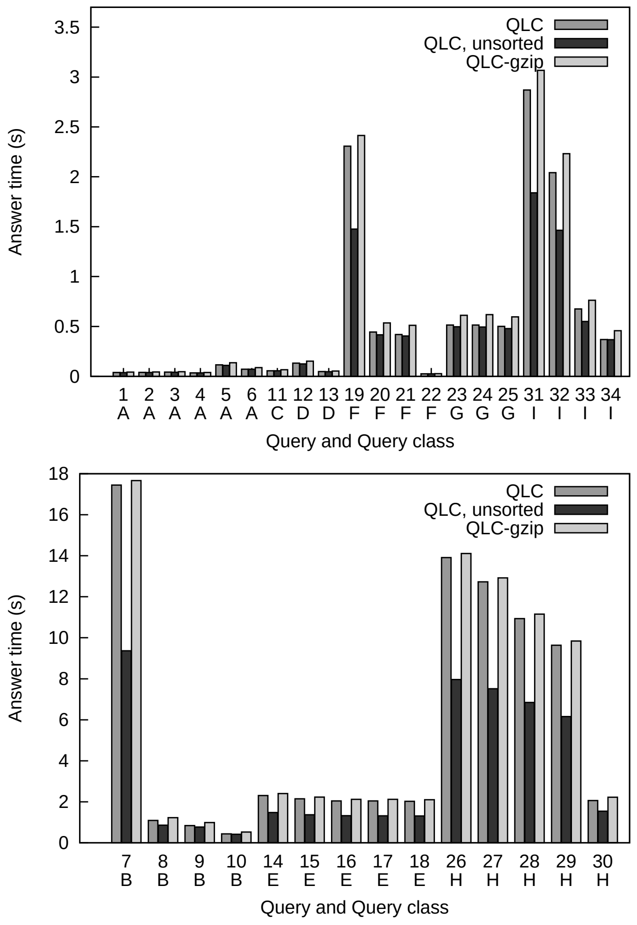


Figure 6.13: Average query times of sorted and unsorted QLC answers and sorted answers on QLC-`gzip` compressed large data. The query classes with small or medium output (*A, C, D, F, G* and I) are on the upper pane and query classes with large output (*B, E* and *H*) on the lower pane.

Returning to the analysis of QLC, the effect of the frequency threshold is interesting (Figure 6.16). With the small frequency threshold values the query times are slightly reducing when the frequency threshold is increased. However, with the large frequency threshold values, the time increases again. The reason is that with small frequency threshold values, there are plenty of compression patterns and the algorithm spends time in matching the query against them and entries in their supports. With large

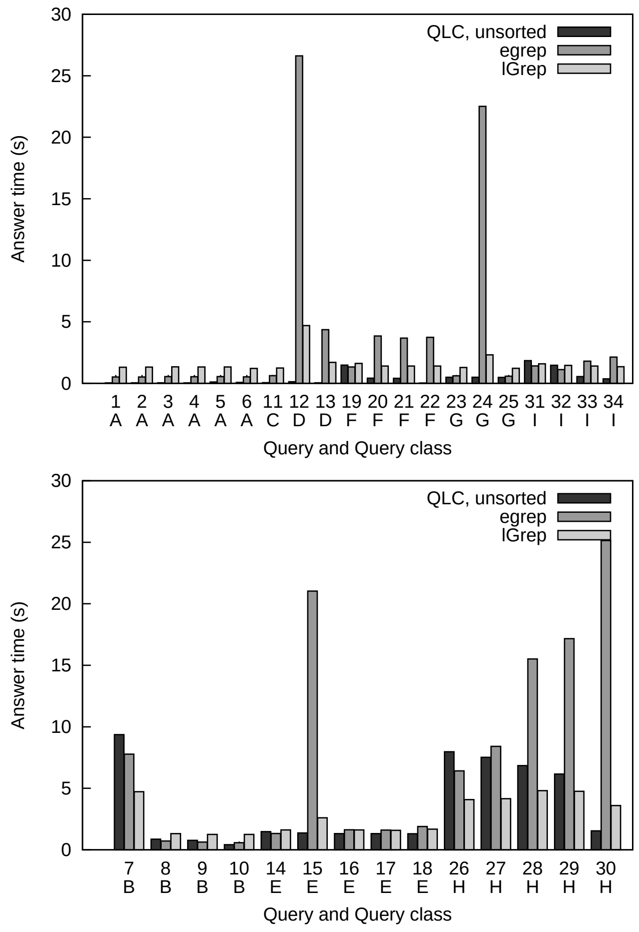


Figure 6.14: Average query times of QLC queries with unsorted answer, `egrep` and *lGrep* on large data. The query classes with small or medium output (*A, C, D, F, G* and I) on the upper pane and query classes with large output (*B, E* and *H*) on the lower pane.

frequency threshold values, the number of compression patterns is small, but the number of unmatched entries is large and the algorithm needs to match the query against each of them.

Experiments with QLC queries show that the method speeds up query evaluation on compressed data, especially when the size of the query answer is less than a few thousand entries. For example, with queries in all the classes resulting in a small or medium answer — i.e., large data queries in

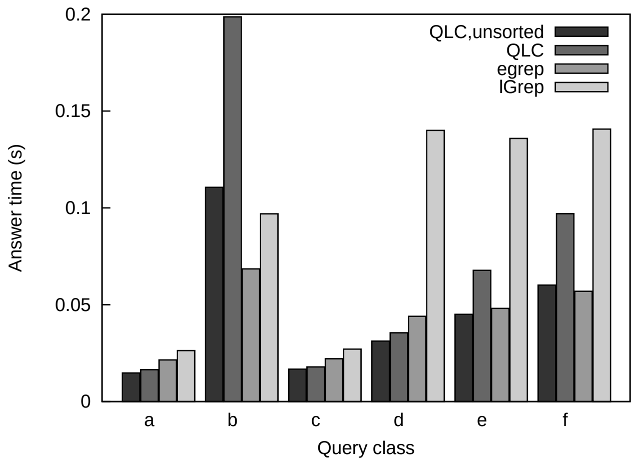


Figure 6.15: Average query times of QLC query with sorted and unsorted answer, `egrep` and *lGrep* by query class of the small data.

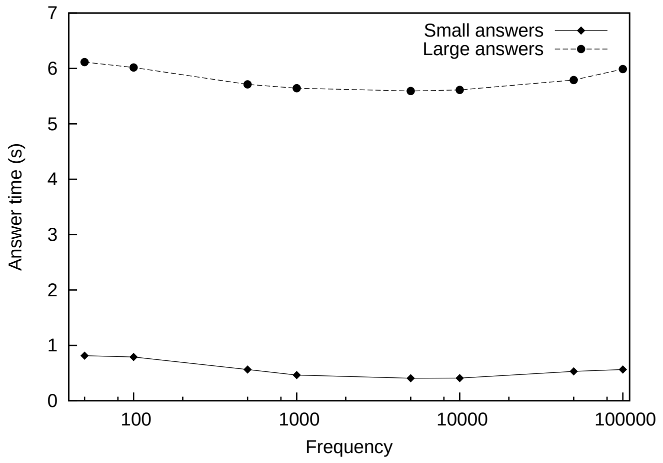


Figure 6.16: Average query times of QLC queries with sorted answers as a function of the frequency threshold. The query classes with small or medium output (*A, C, D, F, G* and I) have been included into small answers and (*B, E* and *H*) into large answers.

classes *A, B, C* and *D* (Table 6.1) and small data queries in classes *a, c, d* and *e* (Table 6.2) — QLC with unsorted answers is the fastest solution.

In classes that result in a large answer — more than several thousands of entries — QLC querying efficiency suffers from the size of the answer. However, it is still feasible; at the longest of all cases a QLC query took less than 40 seconds to execute. The query was Query 7 in large data query class B. It returns an entire log file, i.e., more than $1,100,000$ entries at most.

QLC benefits from adding new *field:value* pairs to queries. For example, as the queries on large data in classes C, D, E, F, G, H and I are enlarged, the execution time reduces (Figure 6.14). The reason for this is that the answer becomes smaller. `egrep` and *lGrep* may react in quite the opposite way; their execution time increases with larger queries, for example, with large data queries in class H.

6.4 Conclusions and related work

QLC compression together with the corresponding query mechanism provide tools to handle the ever-growing problem of log data archiving and analysis. The compression method is efficient, robust, reasonably fast, and when used together with, for example, the `gzip` program provides smaller results than `gzip` alone.

QLC compressed files can be queried without decompressing the whole file. QLC query evaluation is remarkably faster than the combination of decompressing a log file and searching for the corresponding regular expression from the decompressed log; especially when the answer size is less than a few thousand entries.

The query algorithm presented in Figure 6.5 is comparable to decompression: it loops through all the entries and inspects the related pattern and possibly also the remaining items. However, by cashing the results of the intersection between the query and the patterns, the algorithm can minimise the number of needed set comparison or pattern matching operations.

QLC shares many ideas with the CLC method (Chapter 5). Both methods use selected closed sets — filtering and compression patterns — to summarise and compress the data. CLC is a lossy compression method: if an entry is in the support of a filtering pattern, it is removed completely. QLC is lossless: only the values overlapping with the matching compression pattern are removed from an entry and replaced with a reference to the pattern. In QLC compression the main criterion is not understandability as in CLC but storage minimisation and query optimisation. Therefore the criterion for selecting a compression pattern is its compression gain. On the other hand it is a straightforward task to generate a CLC description from a log database compressed with QLC.

As with the CLC, in the data that contain several tens or even hundreds of items in the largest frequent sets, the maximal frequent sets may be a good choise instead of the closed sets. However, as with the CLC, the comparison between closed and maximal frequent sets is left open for further studies.

A related approach that has been published after the original publication of the QLC method [56] also uses frequent patterns for database compression [141, 8]. A method named Krimp [154] takes advantage of minimum description length (MDL) principle [47] in selecting the best set of frequent itemsets, i.e., *"that set that compresses the database best"* [141]. The method uses a greedy heuristic algorithm to search for the best set of frequent patterns. The set is used as a code table to replace parts of database entries that match with a selected frequent item set with shortest possible codes.

The Krimp-method has many advantages: the patterns in the code table can be used in classification [153, 154] or to characterise differences between two data sets [158]. The method does not offer any means for query evaluation on compressed data. However, it could be made with a quite similar algorithm as QLC query evaluation.

All of the four decision subtasks defined in Section 4.2 need log file analysis where archived or otherwise compressed log files are used. In operation of a telecommunications network, system state identification and prediction, cost estimation and estimation of external actions, all require information from logs that the network provides. Especially the security analysis is based on the data recorded in the logs about who did what and when and where and how they came in and how often they used the systems.

The QLC method supports analysis of compressed data on all operation levels. It offers a fast and robust tool to do iterative querying on history data in the knowledge discovery process on the strategic level as well as enables analysis of a recent burst of log entries, which can be kept on the disk only in a compressed format. The method speeds up the iteration by answering queries faster than, for example, the `zcat-egrep` combination commonly used on log files.

QLC versus requirements The QLC method answers well to the requirements set for the data mining and knowledge discovery methods and tools summarised in Section 4.5. The method does not require data-mining-specific knowledge when it is used (*Easy-to-use methods, Easy to learn*).

From a telecommunications expert's point of view, the data mining technology — frequent closed sets — may be integrated into an existing query tool in such a way that the expert does not have to know anything about it (*Interfaces and integrability towards legacy tools*). The technology supports the expert, who can concentrate on identifying queried fields and their most interesting values (*Application domain terminology and semantics used in user interface*). The method shortens the time required to answer queries on archived history data (*Immeadiate, accurate and understandable results, Efficiency and appropriate execution time*). This speeds up the analysis task (*Increases efficiency of domain experts*).

Only the requirements of *Reduced iterations per task*, *Adaptability to process information* and *Use of process information* are not directly addressed. However, the amount of available data and efficiency of an expert are increased. Thus the expert can better concentrate on and take advantage of the information about the network.

Chapter 7

Knowledge discovery for network operations

During the research that began already during the TASA project at the University of Helsinki, one of the main themes has been to bring data mining and knowledge discovery tools and methods to an industrial environment where large amounts of data are being analysed daily. As was described in Chapter 4, the telecommunications operation environment sets requirements that differ from those set by the research environment. The methods presented in Chapters 5 and 6 address many of those industrial requirements.

Based on experiences of industrial applications of the CLC and QLC methods and other data analysis methods on alarm and performance data [59, 61, 60, 69, 70, 97, 96, 58, 63, 64, 98, 102, 103, 156] I will discuss, in this chapter, decision-making and knowledge discovery as a part of an everyday process of network operations [156]. I will outline how the pace and dynamics of this process affects the execution of knowledge discovery tasks, and present an enhanced model for the knowledge discovery process [63]. I will also propose a setup for a knowledge discovery system that better addresses the requirements identified in Chapter 4 [65].

The classical knowledge discovery model (Section 3.1) was designed for isolated discovery projects. As the methods and algorithms presented in this thesis were developed and integrated to network management tools, it became evident that the process model does not fit real-world discovery problems and their solutions. The model needs to be augmented for industrial implementation, where knowledge discovery tasks with short time horizons follow each other continuously. The discovery tasks have to be carried out in short projects or in continuous processes, where the discovery tasks are mixed with other operational tasks.

The main information source for experts operating the networks is the process information system. The system is implemented mainly with legacy applications without data mining functionalities. The knowledge discovery process needs to be integrated into these legacy applications. The integration can be done by including required data mining functionalities to the system. These functionalities have to support network operation experts in their daily work

7.1 Decision making in network operation process

Operation of a mobile telecommunications network is a very quickly evolving business. The technology is developing rapidly; each technology generation lifetime has been less than ten years so far. For example, the benefits of digital technology in the second-generation networks, such as Global System for Mobile communications (GSM), overtook the first-generation analogue systems such as Nordic Mobile Telephony (NMT) in the mid nineties, General Packet Radio Service (GPRS) solutions began to extend GSM networks around 2001 [138], and the third-generation networks such as Universal Mobile Telecommunications System (UMTS) are widely used.

In this environment strategic planning is a continuous activity targeted at the time frame from present to 5 or 10 years. Due to a continuously developing technology base many issues have to be left open when investment decisions are made. While the new technology empowers users with new services, it also changes consumption and communication patterns. The change affects directly the profit achievable through strategic decisions. Hence the effective time horizon of strategic decisions can be as short as one to two years or even less. Their time horizons are shorter than those of some tactical decisions, like planning and executing network infrastucture updates, which can be from two to three years.

In many systems today the redesign cycle is also so rapid that the update of knowledge obtained through knowledge discovery takes time comparable to the redesign cycle time. This creates a swirl in which the strategic and tactical levels can no longer be considered separately. For example, when new network technology is added to the network, the resource planning and redesign of networks are done continuously. All the time some part of the system is under redesign.

At the tactical level of telecommunications network operation there are several continuous maintenance cycles that operate on the same infrastructure. The fastest maintenance cycle is from some seconds to minutes. In it operators try to detect and fix the most devastating malfunctions on the network. The second cycle takes some days, during which operator personnel fix malfunctions that disturb the network capacity and quality of service, but which are not urgent. The next maintenance cycle monitors and audits the network elements and services on a monthly basis. Each component is checked for any malfunctions. If there are needs for configuration optimisations they are also made in this cycle.

The quickly developing technology and continuously evolving set of services also have another consequence: domain knowledge may lose its accuracy rather fast. Configuration parameter updates, installation of new hardware, and updates in existing software components change the structure of the system hence outdating *a priori* knowledge. Constant changes in user behaviour patterns speed this degradation of *a priori* knowledge further.

Knowledge discovery projects — if implemented according to the classical KD-process model (Section 3.1) — typically require quite a long time to execute; at least some months. As most of the tactical decision tasks have a much shorter time horizon, the classical KD-process model can not be applied to their support.

The knowledge discovery efforts supporting these fast-pace strategic and tactical decision-making tasks — *agile decision making* — have to cope with these requirements;

- Decision task time horizons are short
- Decision tasks repeat constantly
- *A priori* knowledge outdates rather fast

These requirements lead to an idea of a continuous knowledge discovery process that finds new information, integrates it to existing knowledge and also evaluates the accuracy of it.

7.2 Knowledge discovery for agile decision support

A decision task always requires information and knowledge about the process and its environment (Section 4.2). If the required information is available, it can be searched for from *a priori* knowledge, otherwise a knowledge discovery task to the available data is needed. The knowledge discovery task is executed in a knowledge discovery process (Figure 7.1).

The execution of a knowledge discovery process uses resources from the existing information systems and applies appropriate data mining and data analysis methods. The results of the knowledge discovery process are interpreted by the expert executing the knowledge discovery task and provided to the decision maker for further consideration.

A knowledge discovery process can be initialised specially for a decision task if it is large and has a long enough time horizon. Typical examples are

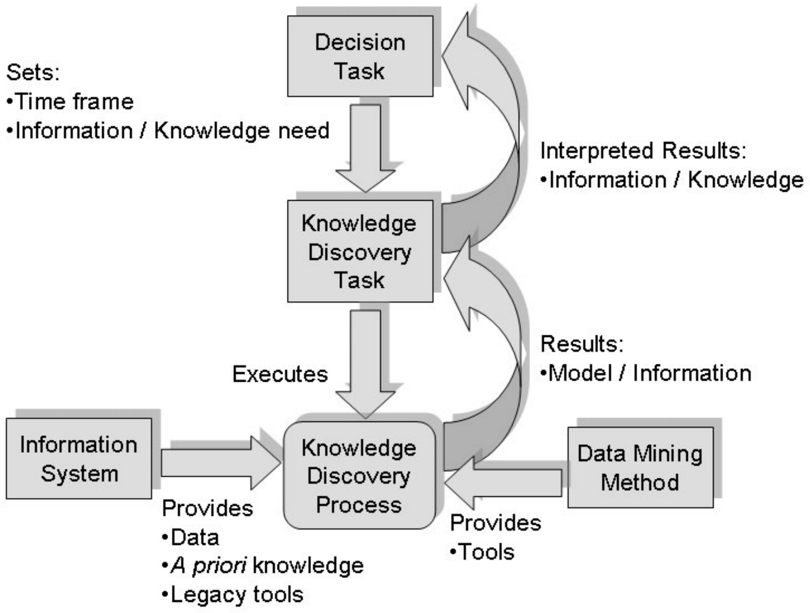


Figure 7.1: Each knowledge discovery task is implemented with a knowledge discovery process utilising the data and legacy tools of process information system and different data mining methods.

knowledge discovery for a strategical decision task or for an expert system making automated process decisions; for example, a search for a covering set of correlation patterns after a major system upgrade.

If the time horizon of a decision task is short, the knowledge discovery task has to be executed in some existing knowledge discovery process. The organisation has to have knowledge discovery tools integrated into the system and available pre-defined processes on how to use them. An example of such a task could be daily log analysis, which searches for security or malfunction incidents that have not been detected and prevented by automatic tools.

Knowledge discovery swirl

The current models for knowledge discovery (see Section 3.1) seem to suffer from the division into phases. The proposed steps are overlapping and it is seldom possible to say where one step ends and another begins [161]. On the other hand, decision tasks often arise sequentially, and for each of them a corresponding knowledge discovery task is introduced as is de-

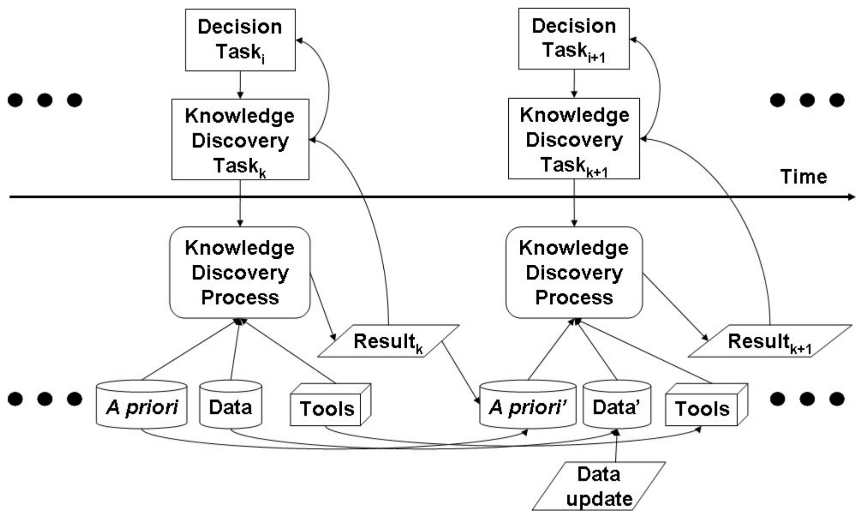

Decision Task$_i$
Knowledge Discovery Task$_k$
Knowledge Discovery Process
Result$_k$
Decision Task$_{i+1}$
Knowledge Discovery Task$_{k+1}$
Time
Knowledge Discovery Process
Result$_{k+1}$
A priori
Data
Tools
A priori'
Data'
Tools
Data update

Figure 7.2: While knowledge discovery tasks are evaluated in a row, they are linking to each other by updating the *a priori* knowledge.

picted in Figure 7.2. Especially if these tasks are executed in one single knowledge discovery process, for example, due to their smaller size or short time horizon, the knowledge discovery tasks link together to form a continuous process, in which each task relies on the results of the previous tasks.

A continuous knowledge discovery process is cyclic: it is a *knowledge discovery swirl* [63] (see Figure 7.3). It is based on a notion of inductive data analysis. Inductive data analysis is a process where data is repeatedly studied and transformed with a collection of methods. Either *a priori* knowledge about the domain, or the results extracted with previous methods, are used for parameterising each method. Each method is applied either to a combination of the results from previous steps or raw data. A method can either extract new information or transform the data into a new form. Every time a new method is selected, a group of functions that are needed for its application are determined and executed.

Selection of accurate, subsequent methods and adjustment of their parameters is affected by the knowledge extracted from the data. The swirl is terminated when the extracted and validated knowledge satisfies the requirement set. If the knowledge is not adequate or new knowledge discovery tasks have arisen, the swirl continues to cycle with new methods and adjusted parameters for previous methods. To validate the results, an expert

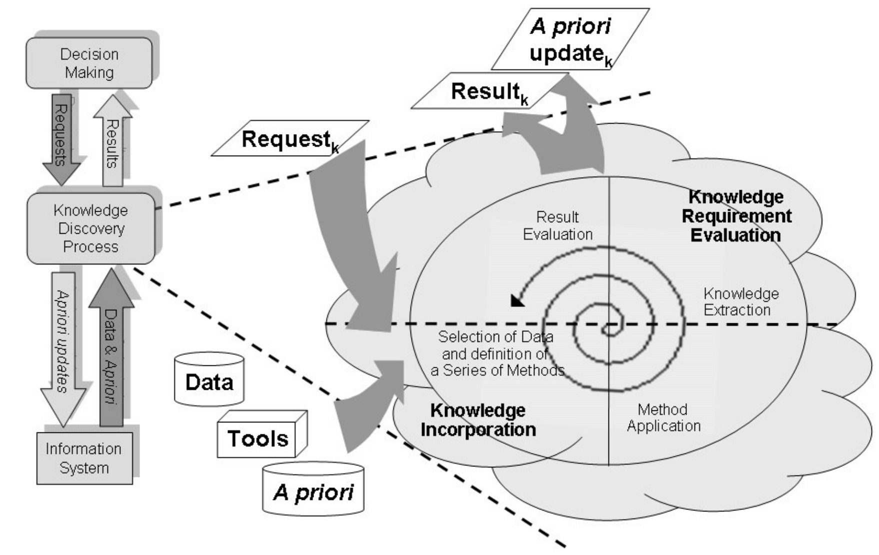


Figure 7.3: Knowledge discovery swirl.

needs appropriate tools that link the results to the data and *a priori* knowledge. The swirl model emphasises the importance of the clearly defined knowledge requirements and facilities to validate the results.

The daily log analysis task, for example, starts already during the system start-up period. During it, the initial set of correlation patterns are generated and deployed. Patterns may filter out entries or replace them with a new entry. This reduces the number of entries, which changes the statistical indicator distributions. The analysis tools have to be tuned according to these changed figures. Every time some new process state is reached, the system may produce a previously unknown, probably voluminous, set of log entries or feature value combinations and the pattern collection must be updated. This is done repeatedly during the system life time either periodically or when needed.

The intention-mining model (I-MIN) [50] gives a similar type of approach to the discovery process as the definition for inductive data analysis. The introduction of the accumulation phase of the model answers particularly to needs of telecommunications network operation. However, unlike the I-MIN process we emphasise the possibility not only to study knowledge discovery concentrates [50] but also to query and analyse any subsets of the original data. The cyclic model also gives a possibility to integrate external knowledge to the process during any of the iterations.

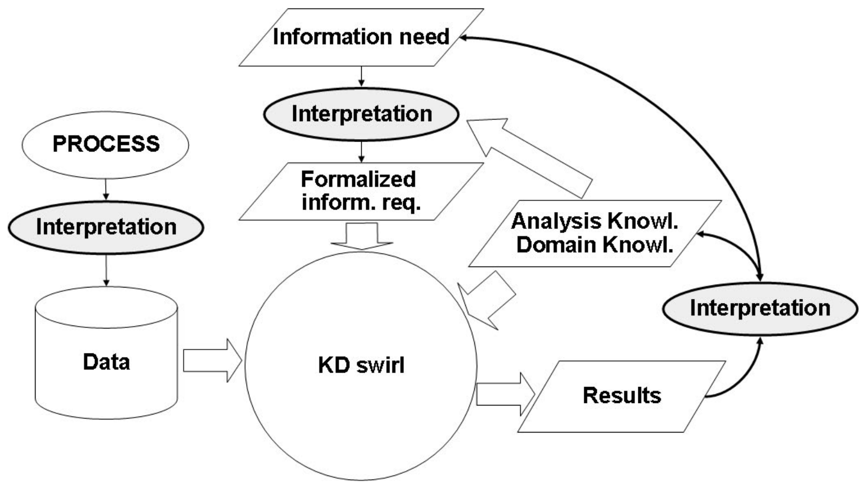


Figure 7.4: Role of interpretations between data and information in knowledge discovery process.

Both of the models have greatly been inspired by the work done within the area of inductive databases [49, 75, 106].

7.3 Adaptive knowledge discovery solution

All the translations between data and knowledge require human interpretation. A human expert makes the interpretation, for example, when he is looking at the available data and making decisions about how the industrial process should be adjusted. In order to be able to make the decisions, he combines the acquired information with his mental model of the subject [105, 31]. He often also verifies his jugdement against the collected history data. This history data must be large and covering. It must also be accessible; such a data set is typically neither easy to store nor simple to query. Here methods like queryable lossless compression QLC (Chapter 6) can assist.

As is depicted in Figure 7.4, there are several places of the information system where interpretation is made between usable information and some formal representation like data, analysis results or formal information request. First of all the log entries are defined in such a way that they carry meaningful information. Knowledge requirements that are defined must be formalised so that the knowledge discovery system is able to find the required information and the knowledge discovery swirl can determine

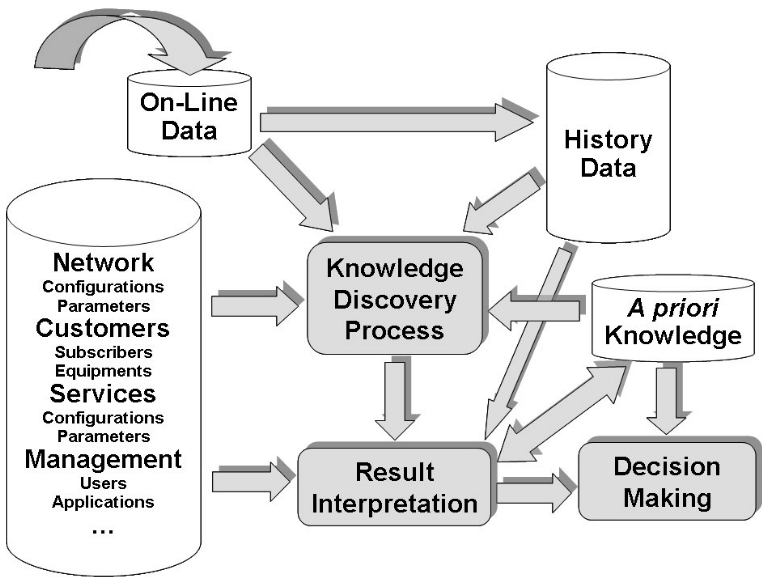


Figure 7.5: An adaptive knowledge discovery process integrated to information system of a network operator.

when the requirements are met. Finally the analysis results revealed in the knowledge discovery swirl need to be interpreted in domain-specific semantic terms so that necessary decisions can be made.

An adaptive knowledge discovery process integrated to an information system of telecommunications network operations is depicted in Figure 7.5. The knowledge discovery process analyses the incoming data. Domain experts doing the analysis use *background information*, i.e., history data, domain configurations and *a priori* knowledge of the organisation to filter the data, to select analysis parameters, to separate causally related phenomena from each other [57] and to steer and focus analysis.

When the information request has been fulfilled, the knowledge discovery process results are interpreted for the decision making. Again all the appropriate background information is used. If the discovery reveals new knowledge that will benefit the organisation, the knowledge is integrated to the *a priori* knowledge. The integration is made only if the benefits of the knowledge considerably exceed the integration cost.

In practice, especially in agile decision making on a tactical level, a large part of the extracted information is never formalised and integrated to the *a priori* knowledge. Reasons vary from haste to obscure or cumbersome knowledge formats. However, experts usually take advantage of that information by learning and improving their own expertise.

The presented model brings the knowledge discovery process to the centre of the process information system given in Figure 4.1. The model connects the knowledge discovery to data interpretation phase. Together with legacy tools, designed for monitoring and interpreting the data, knowledge discovery tools can provide more information and facilitate new insights of experts. This improves also the update and maintenance of *a priori* knowledge. A prerequisite for integration of knowledege discovery process and every day data interpretation is that the knowledge discovery tools fulfil the requirements described in Chapter 4.

The presented knowledge discovery model (Figure 7.5) supports the decision making model introduced in Section 4.2. The knowledge discovery process gets required information about system states, actions and costs from the on-line data, configuration databases and history model. The description of the external world is acquired from network data, i.e., alarms, log entries, traffic measurements, service definitions and subscriber databases of the management system.

7.4 Use of knowledge discovery

On the level of automated process decisions, closed loop control and error detection are widely applied in industrial processes. They often operate in a limited scope for which the KD process and knowledge discovery systems can provide the necessary support. For example, for alarm correlation engines, systems like TASA can be used to extract required knowledge. When methods used in TASA are further augmented to use *a priori* knowledge about the network structure [58], the quality of its results improves.

When the degree of freedom in decision making increases, only a human expert can make decisions. The expert's task starts when many closed control loops are integrated together and the decisions concerning their co-operation have to be made. For this the expert can also be helped by knowledge discovery methods.

When previously unknown industrial process states are analysed, the role of a knowledge discovery system is to filter the needed information and discover structures in it. An expert has to be able to identify the causal relations holding in the logged and measured industrial process. He has to be able to make interpretation and mapping from the log entries and time series to the domain semantics. The methods presented in Chapters 5 and 6 aim at this kind of support. Summarised and compressed logs (CLC and

QLC methods) improve the quality of information that is presented for an expert and increase the amount of available data and information.

An adaptive knowledge discovery process versus requirements The knowledge discovery swirl and its integration to process information model supports well the requirements set for the data mining and knowledge discovery methods and tools summarised in Section 4.5. The model facilitates use of domain specific terminology and semantics by connecting the configuration and parameter information to the analysis process (*Application domain terminology and semantics used*, *easy to learn*). It also supports linking the analysis results to related process information (*Use of process information*). Thus analysis results can be described directly in system context and interpretation of results becomes easier (*Easy-to-use methods*, *Easy to learn*). This simplifies integration of data mining and legacy tools (*Interfaces and integrability towards legacy tools*) and makes it possible for analysis methods to adapt current process information (*Use of process information*, *Adaptability to process information*). The model facilitates the shortening of the time required to perform the analysis tasks and interprete their results (*Immeadiate, accurate and understandable results*, *Efficiency and appropriate execution time*, *Increases efficiency of domain experts*). Only the requirement of *Reduced iterations per task* is left for data mining tools.

Chapter 8

Concluding remarks

In this thesis I have studied application of data mining and knowledge discovery methods and processes in telecommunications network operations. The objective of the thesis was to find ways to assist operators in solving everyday problems in decision-making.

First, I studied the telecommunications operations as an environment for data mining and knowledge discovery methods and tools (Chapter 4). I introduced an operation information system and a decision-making model, discussed decision making on different operational levels and studied decision-making resources and assets available for used methods and tools.

When the requirements and resources for data mining applications were defined, I introduced data summarisation (Chapter 5) and compression (Chapter 6) methods based on closed sets and made an extensive set of experiments with both methods. Finally, I defined a continuous knowledge discovery swirl and outlined how that can be implemented to support knowledge extraction for repeating decision tasks from a continuous data flow in a telecommunications operation environment (Chapter 7).

The methods introduced, especially CLC and its generalisation to episode summarisation, have been accepted well among network experts. CLC, for example, has been integrated to the security log management and analysis software called NetAct $^{\mathrm{TM}}$Audit Trail by Nokia Siemens Networks. The use of domain structure data to improve the quality of episode summaries [58] was one of the enablers for the acceptance.

In this thesis I have focused on log data. Another important type of data are the measurement and counter time series that are continuously collected from the network elements. An interesting question is what kind of knowledge discovery methods could provide corresponding benefits for time series data analysis. Unsupervised clustering and anomaly detection seem to be promising candidates. Furthermore, how could one combine information extracted from both types of data? These questions are currently under my consideration.

The results of this thesis and their acceptance among telecommunications experts show that the approach to support human expertise works in practice. As has been shown elsewhere [123], expert systems and control loops are able to automate decisions that are either very frequent and almost trivial, or hard but well defined. In other cases, especially in tactical decision making, the more productive way is to prepare to support the cognition of a human expert. At the tactical level we need support for on-line decision making, which is quite a challenge for many of the knowledge discovery methods and tools. The tools should provide an easy-to-use approach with minimal iterations that reveal understandable results for experts in the domain. The method should be robust, provide accurate and reliable information and — what is most important — help the domain experts in their daily work. This is exactly what the CLC and QLC methods do.

References

[1] Douglas Aberdeen. A (revised) survey of approximate methods for solving partially observable Markov decision. Technical report, National ICT Australia, 2003.

[2] Foto Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In Ronny Kohavi, Johannes Gehrke, William DuMouchel, and Joydeep Ghosh, editors, *Proceedings of the tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 12 – 19, Seattle, Washington, USA, August 2004. ACM.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, Washington, D.C., USA, May 1993. ACM.

[4] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, California, USA, 1996.

[5] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison Wesley, Reading, Massachusets, USA, 1986.

[6] Shivnath Babu, Minos Garofalakis, and Rajeev Rastogi. SPARTAN: A model-based semantic compression system for massive data tables. In Walid G. Aref, editor, *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 283 – 294, Santa Barbara, California, USA, May 2001. ACM.

[7] Raju Balakrishnan and Ramendra K. Sahoo. Lossless compression for large scale cluster logs. Technical Report RC23902(W0603-038), IBM Research Division, March 2006.

[8] Ronnie Bathoorn, Arne Koopman, and Arno Siebes. Frequent patterns that compress. Technical Report UU-CS-2006-048, Utrecht University, Department of Information and Computing Sciences, Utrecht, The Netherlands, 2006.

[9] Roberto J. Bayardo. Efficiently mining long patterns from databases. In Laura Haas and Ashutosh Tiwary, editors, *SIGMOD 1998, Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 85 – 93, Seattle, Washington, USA, June 1998. ACM Press.

[10] Richard J. Bolton and David J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17(3):235 – 255, 2002.

[11] Francesco Bonchi, Fosca Giannotti, and Dino Pedreschi. A relational query primitive for constraint-based pattern mining. In Jean-François Boulicaut, Luc De Raedt, and Heikki Mannila, editors, *Constraint Based Mining and Inductive Databases*, volume 3848 of *LNCS*, pages 14 – 37. Springer, 2006.

[12] Jean-François Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In Takao Terano, Huan Liu, and Arbee L.P. Chen, editors, *Proceedings of the Fourth Pacific- Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2000)*, volume 1805 of *LNAI*, pages 62–73, Kyoto, Japan, April 2000. Springer.

[13] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Approximation of frequency queries by mean of free-sets. In Djamel A. Zighed, Jan Komorowski, and Jan Zytkow, editors, *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000*, volume 1910 of *LNAI*, pages 75 – 85, Lyon, France, September 2000. Springer.

[14] Jean-François Boulicaut, Artur Bykowski, and Christophe Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery journal*, 7(1):5 – 22, 2003.

[15] Jean-François Boulicaut and Baptiste Jeudy. Mining free-sets under constraints. In Michel E. Adiba, Christine Collet, and Bipin C. Desai, editors, *Proceedings of International Database Engineering & Applications Symposium (IDEAS'01)*, pages 322 – 329, Grenoble, France, July 2001. IEEE Computer Society.

[16] Jean-François Boulicaut and Baptiste Jeudy. Constraint-based data mining. In Oded Maimon and Lior conRokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 399 – 416. Springer, 2005.

[17] Ronald Brachman and Tej Anand. The process of knowledge discovery in databases. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 37 – 58. AAAI Press, Menlo Park, California, USA, 1996.

[18] Ronald Brachman, Tom Khabaza, Willi Klösgen, Gregory Piatetsky-Shapiro, and Evangelis Simoudis. Mining business databases. *Communications of the ACM*, 39(11):42 – 48, November 1996.

[19] Leo Breiman, Jerome Friedman, R. A. Olshen, and Charles J. Stone. *Classification and Regression Trees.* Chapman & Hall CRC Press LLC, Boca Raton, Florida, USA, 1984.

[20] T. Bui, K. Higa, V. Sivakumar, and J. Yen. Beyond telecommuting: Organizational suitability of different modes of telework. In J.F. Nunamaker and R.H. Sprague, editors, *Proceedings of the 29th Annual Hawaii International Conference on System Sciences*, pages 344 – 353, Maui, Hawaii, USA, January 1996. IEEE Computer Society Press.

[21] Doug Burdick, Manuel Calimlim, and Johannes Gehrke. MAFIA: A maximal frequent itemset algorithm for transactional databases. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'2001)*, pages 443 – 452, Heidelberg, Germany, April 2001. IEEE Computer Society.

[22] M. Burrows and D. J. Wheeler. A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, Palo Alto, California, USA, May 1994.

[23] Artur Bykowski and Christophe Rigotti. A condensed representation to find frequent patterns. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'01)*, pages 267 – 273, Santa Barbara, California, USA, May 2001. ACM Press.

[24] Toon Calders and Bart Goethals. Mining all non derivable frequent itemsets. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen,

editors, *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002*, volume 2431 of *LNAI*, pages 74 – 83, Helsinki, Finland, August 2002. Springer.

[25] Toon Calders, Christophe Rigotti, and Jean-François Boulicaut. A survey on condensed representations for frequent sets. In Jean-François Boulicaut, Luc De Raedt, and Heikki Mannila, editors, *Constraint Based Mining and Inductive Databases*, volume 3848 of *LNCS*, pages 64 – 80. Springer, 2006.

[26] Varun Chandola and Vipin Kumar. Summarization – compressing data into an informative representation. In *Proceedings of the fifth IEEE International Conference on Data Mining (ICDM '05)*, pages 98 – 105, Houston, Texas, USA, November 2005. IEEE Computer Society Press.

[27] Varun Chandola and Vipin Kumar. Summarization – compressing data into an informative representation. *Knowledge and Information Systems*, 12(3):355 – 378, August 2007.

[28] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet Firewalls.* O'Reilly & Associates, 1995.

[29] William R. Cheswick, Steven M. Bellovin, and Aviel D. Rubin. *Firewalls and Internet Security.* Addison Wesley, 2003. 2nd edition.

[30] Theresa Chung and Steve Luc, editors. *Check Point Security Administration NGX I, Student Handbook.* Checkpoint Software Technologies Ltd., 2006.

[31] James Courtney. Decision making and knowledge management in inquireing organizations: toward a new decision -making paradigm for DSS. *Decision Support Systems*, 31(1):17 – 38, 2001.

[32] Robert G. Cowell, A. Philip Dawid, Steffen L. Lauritzen, and David J. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Statistics for Engineering and Information Science. Springer-Verlag, Berlin, 1999.

[33] Bruno Crémilleux and Jean-François Boulicaut. Simplest rules characterizing classes generated by delta-free sets. In Max Bramer, Alun Preece, and Frans Coenen, editors, *Proceedings of the twenty-second SGAI International Conference on Knowledge Based Systems and Applied Artificial Intelligence, ES 2002*, pages 33 – 46, Cambridge, United Kingdom, December 2002. Springer.

[34] Gautam Das, K.-I. Lin, Heikki Mannila, G Renganathan, and Padraic Smyth. Rule discovery from time series. In Rakesh Agrawal, Paul Stolorz, and Gregory Piatetsky-Shapiro, editors, *Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 16 – 22, New York, USA, August 1998. AAAI Press.

[35] Ann Devitt, Joseph Duffin, and Robert Moloney. Topographical proximity for mining network alarm data. In *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, pages 179 – 184, Philadelphia, Pennsylvania, USA, August 2005. ACM.

[36] Milenko Drinić and Darko Kirovski. PPMexe: PPM for compressing software. In *Proceedings of 2002 Data Compression Conference (DCC 2002)*, pages 192 – 201, Snowbird, Utah, USA, April 2002. IEEE Computer Society.

[37] Ian Eaton. *The Ins and Outs of System Logging Using Syslog.* InfoSec Reading room. SANS Institute, 2003.

[38] European telecommunication standards institute ETS 300 616. Digital cellular telecommunications system (Phase 2); Event and call data (GSM 12.05 version 4.3.1). European telecommunication standards institute, February 1998.

[39] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1 – 34. AAAI Press, Menlo Park, California, USA, 1996.

[40] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, California, USA, 1996.

[41] E. Fredkin. Trie memory. *Communications of the ACM*, (3):490 – 500, 1960.

[42] Simon French and David Rios-Insua. *Statistical Decision Theory.* Kendall's Library of Statistics. Arnold Publishing, 2000.

[43] Simson Garfinkel and Gene Spafford. *Practical Unix & Internet Security, Second edition.* O'Reilly & Associates, 1996.

[44] Minos N. Garofalakis and Rajeev Rastogi. Data mining meets network management: The NEMESIS project. In *Online proceedings of ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, DMKD 2001*, Santa Barbara, California, USA, May 2001. ACM.

[45] Gartner says worldwide mobile phone sales increased 16 per cent in 2007. Press Release. Gartner Inc., February 2008.

[46] G.Anthony Gorry and Michael S. Scott Morton. A framework for management information systems. *Sloan Management Review*, 13(1):50 – 70, 1971.

[47] Peter D. Grünwald. Minimum description length tutorial. In Peter D. Grünwald, In Jae Myung, and Mark A. Pitt, editors, *Advances in minimum description length*, pages 23 – 80. MIT press, Cambridge, Massachusets, USA, April 2005.

[48] Dimitrios Gunopulos, Heikki Mannila, and Sanjeev Saluja. Discovering all most specific sentences by randomized algorithms. In Foto N. Afrati and Phokion G. Kolaitis, editors, *Database Theory - ICDT '97, 6th International Conference*, volume 1186 of *LNCS*, pages 215 – 229, Delphi, Greece, January 1997. Springer.

[49] S.K. Gupta, V. Bhatnagar, and S. K. Wasan. User centric mining of association rules. In Alipio M. Jorge and Pavel Brazdil, editors, *Proceedings of Workshop on data mining, decision support, meta learning and ILP (DDMI'00) co-located with PKDD'00*, Lyon, France, September 2000.

[50] S.K. Gupta, V. Bhatnagar, S. K. Wasan, and DVLN Somayajulu. Intension mining: A new paradigm in knowledge discovery. Technical Report IITD/CSE/TR2000/001, Indian Institute of Technology, Department of Computer Science and Engineering, New Delhi 110 016, India, March 2000.

[51] John Hadden, Ashutosh Tiwari, Rajkumar Roy, and Dymitr Ruta. Computer assisted customer churn management: state of the art and future trends. *Journal of Computers and Operations Research*, 34(10):2902 – 2917, 2007.

[52] Jiawei Han, Yandong Cai, and Nick Cercone. Knowledge discovery in databases: an attribute-oriented approach. In Le-Yan Yuan, editor, *Proceedings of the Eightteenth International Conference on Very*

Large Data Bases (VLDB'92), pages 547 – 559, Vancouver, British Columbia, Canada, August 1992. Morgan Kaufmann.

[53] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and techniques.* Morgan Kaufmann Publishers, San Francisco, California, USA, 2000.

[54] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining.* The MIT Press, Cambridge, Massachusets, USA, 2001.

[55] Kimmo Hätönen, Jean-François Boulicaut, Mika Klemettinen, Markus Miettinen, and Cyrille Masson. Comprehensive log compression with frequent patterns. In Yahiko Kambayashi, Mukesh K. Mohania, and Wolfram Wöß, editors, *Proceedings of Data Warehousing and Knowledge Discovery, 5th International Conference, (DaWaK 2003)*, volume 2737 of *LNCS*, pages 360 – 370, Prague, Czech Republic, September 2003. Springer.

[56] Kimmo Hätönen, Perttu Halonen, Mika Klemettinen, and Markus Miettinen. Queryable lossless log database compression. In Jean-François Boulicaut and Sašo Džeroski, editors, *Proceedings of th 2nd International Workshop on Knowledge Discovery in Inductive Databases - (KDID'03)*, pages 70 – 79, Cavtat-Dubrovnik, Croatia, September 2003. Ruder Boskovic Institute, Bijenicka cesta 54, P.O.B. 180 Zagreb, Croatia.

[57] Kimmo Hätönen and Mika Klemettinen. Domain structures in filtering irrelevant frequent patterns. In Mika Klemettinen and Rosa Meo, editors, *Proceedings of the First International Workshop on Inductive Databases (KDID)*, pages 50 – 60, Helsinki, Finland, August 2002. Helsinki University Printing House, Helsinki.

[58] Kimmo Hätönen and Mika Klemettinen. Domain structures in filtering irrelevant frequent patterns. In Pier Luca Lanzi, Rosa Meo, and Mika Klemettinen, editors, *Database support for data mining applications*, volume 2682 of *LNCS*, pages 289 – 305. Springer, 2004.

[59] Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Knowledge discovery from telecommunication network alarm databases. In Stanley Y. W. Su, editor, *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, pages 115 – 122, New Orleans, Louisiana, USA, February 1996. IEEE Computer Society Press.

[60] Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. Rule discovery in alarm databases. Technical Report C-1996-7, University of Helsinki, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland, March 1996.

[61] Kimmo Hätönen, Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, and Hannu Toivonen. TASA: Telecommunication alarm sequence analyzer, or "How to enjoy faults in your network". In *Proceedings of NOMS'96 - IEEE Network Operations and Management Symposium (NOMS'96)*, pages 520 – 529, Kyoto, Japan, April 1996. IEEE.

[62] Kimmo Hätönen, Mika Klemettinen, and Markus Miettinen. Remarks on the industrial application of inductive database technologies. In Jean-François Boulicaut, Heikki Mannila, and Luc de Raedt, editors, *Constraint-Based Mining and Inductive Databases*, volume 3848 of *LNCS*, pages 196 – 215. Springer, 2006.

[63] Kimmo Hätönen, Pekka Kumpulainen, and Pekko Vehviläinen. Pre- and post-processing for mobile network performance data. In Reijo Tuokko, editor, *Proceedings of seminar days of Finnish Society of Automation, (Automation 03)*, pages 311 – 316, Helsinki, Finland, September 2003. Finnish Society of Automation.

[64] Kimmo Hätönen, Sampsa Laine, and Timo Similä. Using the logsig-function to integrate expert knowledge to self-organizing map (SOM) based analysis. In *Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications*, pages 145 – 150, Binghamton, New York, USA, June 2003. IEEE.

[65] Kimmo Hätönen, Risto Ritala, and Pekko Vehviläinen. Knowledge discovery in dynamic decision support: Linking strategy and tactics. 2006. (manuscript).

[66] Gisela Hertel. Operation and maintenance. In Friedhelm Hillebrand, editor, *GSM and UMTS. The Creation of Global Mobile Communication*, pages 445 – 456. John Wiley & Sons, Ltd, Chichester, England, 2002.

[67] Friedhelm Hillebrand, editor. *GSM and UMTS. The Creation of Global Mobile Communication.* John Wiley & Sons, Ltd, Chichester, England, 2002.

[68] Johan Himberg. *From insights to innovations: data mining, visualization, and user interfaces.* PhD thesis, Helsinki University of Technology, Laboratory of Computer and Information Science, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland, November 2004.

[69] Albert Höglund and Kimmo Hätönen. Computer network user behaviour visualisation using self organising maps. In Lars Niklasson, Mikael Bodén, and Tom Ziemke, editors, *Proceedings of ICANN98, Eight International Conference on Artificial Neural Networks*, pages 899 – 904, Skövde, Sweden, September 1998. Springer.

[70] Albert J. Höglund, Kimmo Hätönen, and Antti Sorvari. A computer host-based user anomaly detction system using the self-organizing map. In Shun-Ichi Amari, C. Lee Giles, Marco Gori, and Vincenzo Piuri, editors, *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000*, pages 24 – 27, Como, Italy, July 2000. IEEE Computer Society Press.

[71] Jaakko Hollmén. *User Profiling and Classification for Fraud Detection in Mobile Communications Networks.* PhD thesis, Helsinki University of Technology, Laboratory of Computer and Information Science, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland, December 2000.

[72] Jaakko Hollmén, Jouni K. Seppänen, and Heikki Mannila. Mixture models and frequent sets: combining global and local methods for 0-1 data. In Daniel Barbará and Chandrika Kamath, editors, *Proceedings of the Third SIAM International Conference on Data Mining*, pages 289–293, San Francisco, California, USA, May 2003. SIAM.

[73] Jaakko Hollmén and Jarkko Tikka. Compact and understandable descriptions of mixture of bernoulli distributions. In Michael R. Berthold, John Shawe-Taylor, and Nada Lavrač, editors, *Proceedings of the 7th International Symposium on Intelligent Data Analysis (IDA 2007)*, volume 4723 of *LNCS*, pages 1–12, Ljubljana, Slovenia, September 2007. Springer.

[74] Peter Hoschka and Willi Klösgen. A support system for interpreting statistical data. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 325 – 345. AAAI Press, Menlo Park, California, USA, 1991.

[75] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58 – 64, November 1996.

[76] ITU-T, telecommunication standardization sector of ITU. *Enhanced Telecom Operations Map (eTOM) - Introduction.* Number M.3050.0 in ITU-T recommendations. International Telecommunication Union (ITU), March 2007.

[77] ITU-T, telecommunication standardization sector of ITU, Study group 4. *Maintenance philosophy for telecommunication networks.* Number M.20 in ITU-T recommendations. International Telecommunication Union (ITU), October 1992.

[78] ITU-T, telecommunication standardization sector of ITU, Study group 4. *Overview of TMN recommendations.* Number M.3000 in ITU-T recommendations. International Telecommunication Union (ITU), February 2000.

[79] ITU-T, telecommunication standardization sector of ITU, Study group 4. *Principles for a telecommunications management network.* Number M.3010 in ITU-T recommendations. International Telecommunication Union (ITU), February 2000.

[80] ITU-T, telecommunication standardization sector of ITU, study group 4. *TMN management functions.* Number M.3400 in ITU-T recommendations. International Telecommunication Union (ITU), February 2000.

[81] ITU-T, telecommunication standardization sector of ITU, Study group 4. *Security for the management plane: Overview.* Number M.3016.0 in ITU-T recommendations. International Telecommunication Union (ITU), May 2005.

[82] Gabriel Jakobson and Mark Weissman. Real-time telecommunication network management: extending event correlation with temporal constraints. In Adarshpal S. Sethi, Yves Raynaud, and Fabienne Faure-Vincent, editors, *Proceedings of the fourth international symposium on Integrated network management IV*, pages 290 – 301, London, United Kingdom, 1995. Chapman & Hall.

[83] Gabriel Jakobson and Mark D. Weissman. Alarm correlation. *IEEE Network*, 7(6):52 – 59, November 1993.

[84] Heikki Jokinen, Kimmo Konkarikoski, Petteri Pulkkinen, and Risto Ritala. Operations' decision making under uncertainty: case studies on papermaking. *Mathematical and Computer Modelling of Dynamical Systems*, X(X):000–000, Month 200X. Submitted.

[85] Heikki Jokinen and Risto Ritala. Value assessment of measurements in large measurement information systems. In Eleftherios Kayafas and Vassilis Loumos, editors, *Technical Committee on Measurement of Electrical Quantities. Proceedings of the 13th International Symposium on Measurements for Research and Industry Applications and the 9th European Workshop on ADC Modelling and Testing*, volume 1, pages 367 – 372, Athens, Greece, September 2004. IMECO and NTUA.

[86] Leslie Pack Kaelbing, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[87] P. Keen. Information systems and organizational change. *Communications of the ACM*, 24(1):24 – 33, 1981.

[88] Mika Klemettinen. *Rule Discovery from Telecommunication Network Alarm Databases.* PhD thesis, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland, January 1999.

[89] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 401 – 407, Gaithersburg, Maryland, USA, November 1994. ACM.

[90] Willi Klösgen and Jan Zytkow. Knowledge discovery in databases terminology. In Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 573 – 592. AAAI Press, Menlo Park, California, USA, 1996.

[91] Donald Knuth. The art of computer programming. In *Volume 3: Sorting and Searching.* Addison-Wesley, Reading, Massachusets, USA, 1973.

[92] Teuvo Kohonen, editor. *Self-Organizing Maps.* Springer, Berlin, Germany, 1995.

[93] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 2(1):1 – 15, 2000.

[94] Dudyala Anil Kumar and V. Ravi. Predicting credit card customer churn in banks using data mining. *International Journal on Data Analysis Techniques and Strategies*, 1(1):4 – 28, August 2008.

[95] Pekka Kumpulainen and Kimmo Hätönen. Local anomaly detection for network system log monitoring. In M. Konstantionis and I. Lazaros, editors, *Proceedings of the 10th International Conference on Engineering Applications of Neural Networks.*, pages 34 – 44, Thessaloniki, Greece, August 2007. Publishing Centre, Alexander Technological Educational Institute of Thessaloniki.

[96] Pekka Kumpulainen and Kimmo Hätönen. Compression of cyclic time series data. In Eric Benoit, editor, *Proceedings of the 12th IMEKO TC1 & TC7 Joint Symposium on Man Science & Measurement.*, pages 413 – 419, Annecy, France, September 2008. IMEKO.

[97] Pekka Kumpulainen and Kimmo Hätönen. Local anomaly detection for mobile network monitoring. *Information Sciences*, (178):3840 – 3859, 2008.

[98] Mikko Kylväjä, Kimmo Hätönen, and Pekka Kumpulainen. Information summarization for network performance management. In M. Laszlo and J.V. Zsolt, editors, *Proceedings of the 10th TC-10 IMEKO Conference on Technical Diagnostics*, pages 167 – 172, Budapest, Hungary, June 2005. IMEKO.

[99] Mikko Kylväjä, Kimmo Hätönen, Pekka Kumpulainen, Jaana Laiho, Pasi Lehtimäki, Kimmo Raivio, and Pekko Vehviläinen. Trial report on self-organizing map based analysis tool for radio networks. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC 2004 Spring)*, pages 2365 – 2369, Milan, Italy, May 2004. IEEE.

[100] Jaana Laiho, Anneli Korteniemi, Markus Djupsund, Mikko Toivonen, and Jochen Grandell. Radio network optimisation process. In Jaana Laiho, Achim Wacker, and Tomáš Novosad, editors, *Radio Network Planning and Optimisation for UMTS*, pages 329 – 364. John Wiley & Sons, Ltd, Chichester, England, 2002.

[101] Jaana Laiho, Mikko Kylväjä, and Albert Höglund. Utilization of advanced analysis methods in UMTS networks. In *Proceedings of the IEEE Semiannual Vehicular Technology Conference (VTC 2002 Spring) Spring*, pages 726 – 730, Birmingham, Alabama, USA, May 2002. IEEE.

[102] Jaana Laiho, Kimmo Raivio, Pasi Lehtimäki, Kimmo Hätönen, and Olli Simula. Advanced analysis methods for 3G cellular networks. Technical Report A65, Computer and Information Science, Helsinki University of Technology, Espoo, Finland, 2002.

[103] Jaana Laiho, Kimmo Raivio, Pasi Lehtimäki, Kimmo Hätönen, and Olli Simula. Advanced analysis methods for 3G cellular networks. *IEEE Transactions on Wireless Communications*, 4(3):930 – 942, May 2005.

[104] Jaana Laiho, Achim Wacker, and Tomáš Novosad, editors. *Radio Network Planning and Optimisation for UMTS.* John Wiley Inc., Chichester, England, 2006. 2nd edition.

[105] Sampsa Laine. *Using visualization, variable selection and feature extraction to learn from industrial data.* PhD thesis, Helsinki University of Technology, Laboratory of Computer and Information Science, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland, September 2003.

[106] Pier Luca Lanzi, Rosa Meo, and Mika Klemettinen, editors. *Database support for data mining applications*, volume 2682 of *LNCS*. Springer, 2004.

[107] H Leavitt. Applying organizational change in industry: Structural, technological and humanistic approaches. In J. March, editor, *Handbook of Organizations.* Rand McNally, Chicago, Illinois, USA, 1965.

[108] Pasi Lehtimäki. *Data Analysis Methods for Cellular Network Performance Optimization.* PhD thesis, Helsinki University of Technology, Faculty of Information and Natural Sciences, Department of Information and Computer Science, P.O.Box 5400, FI-02015 TKK, Espoo, Finland, April 2008.

[109] Pasi Lehtimäki, Kimmo Raivio, and Olli Simula. Mobile radio access network monitoring using the self-organizing map. In Michel Verleysen, editor, *Proceedings of 10th European Symposium on Artificial Neural Networks (ESANN 2002)*, pages 231 – 236, Bruges, Belgium, April 2002.

[110] Hartmut Liefke and Dan Suciu. XMill: an efficient compressor for XML data. In Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 153 – 164, Dallas, Texas, USA, June 2000. ACM.

[111] Dao-I Lin and Zvi M. Kedem. Pincer-Search: A new algorithm for discovering the maximum frequent set. In Hans-Jörg Schek, Fèlix Saltor, Isidro Ramos, and Gustavo Alonso, editors, *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technologyc*, volume 1377 of *LNCS*, pages 105 – 119, Valencia, Spain, 1998. Springer.

[112] William S. Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1-4):47 – 66, 1991.

[113] Stuart Madnick. From VLDB to VMLDB (very MANY large data bases): Dealing with large-scale semantic heterogeneity. In Umeshwar Dayal, Peter M.D. Gray, and Shojiro Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 11 – 16, Zürich, Switzerland, September 1995. Morgan Kaufmann.

[114] Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146 – 151, Portland, Oregon, USA, August 1996. AAAI Press.

[115] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Improved methods for finding association rules. Technical Report C-1993-65, University of Helsinki, Department of Computer Science, P.O. Box 26, FIN-00014 University of Helsinki, Finland, December 1993.

[116] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192, Seattle, Washington, USA, July 1994. AAAI Press.

[117] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovering frequent episodes in sequences. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International*

Conference on Knowledge Discovery and Data Mining (KDD'95), pages 210 – 215, Montreal, Canada, August 1995. AAAI Press.

[118] Ian I. Mitroff and Harold A. Linstone. *The Unbounded Mind: Breaking the Chains if Traditional Business Thinking.* Oxford University Press, New York, USA, 1993.

[119] Michel Mouly and Marie-Bernadette Pautet. *The GSM System for Mobile Communications.* Cell & Sys, Palaiseau, France, 1992.

[120] Eric W.T. Ngai, Li Xiu, and D.C.K. Chau. Application of data mining techniques in customer relationship management: A literature review and classification. *Expert Systems with Applications*, 36(2):2592 – 2602, March 2009.

[121] Mark Nicolett. *Critical Capabilities for Security Information and Event Management Technology, 2008.* Number G00156303 in Gartner RAS Core Research Notes. Gartner, Inc., May 2008.

[122] F. Niederman and J. Trower. Industry influence on IS personnel and roles. In Mohan R. Tanniru, editor, *Proceedings of the 1993 Conference on Computer Personnel Research (SIGCPR '93)*, pages 226 – 233, St Louis, Missouri, USA, 1993. ACM.

[123] Jukka Nurminen, Olli Karonen, and Kimmo Hätönen. What makes expert systems survive over 10 years - empirical evaluation of several engineering applications. *Journal of Expert Systems with Applications*, 24(3):199 – 211, 2003.

[124] Jukka K. Nurminen. *Modelling and implementation issues in circuit and network planning tools.* PhD thesis, Helsinki University of Technology, Systems Analysis Laboratory, P.O.Box 1100, FIN-02015 HUT, Espoo, Finland, May 2003.

[125] J. Oksanen, H. Paunonen, E. Sipilä, and A. Kaunonen. Improved knowledge management with electronic operating and maintenance manuals. *PaperAge Magazine*, January 2001.

[126] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Closed set based discovery of small covers for association rules. In Christine Collet, editor, *Proceedings of BDA'99*, pages 361 – 381, Bordeaux, France, October 1999.

[127] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski, and Wojciech Ziarko. Rough sets. *Communications of the ACM*, 38(11):88 – 95, November 1995.

[128] Jian Pei, Jiawei Han, and Runying Mao. CLOSET an efficient algorithm for mining frequent closed itemsets. In Dimitrios Gunopulos and Rajeev Rastogi, editors, *Proceedings of 2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21 – 30, Dallas, Texas, USA, May 2000. ACM.

[129] Jyrki Penttinen. *GSM-tekniikka: järjestelmän toiminta ja kehitys kohti UMTS-aikakautta*. WSOY, Helsinki, Finland, 2001. (in Finnish).

[130] Gregory Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In Gregory Piatetsky-Shapiro and William J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229 – 248. AAAI Press, Menlo Park, California, USA, 1991.

[131] Gregory Piatetsky-Shapiro and Christopher J. Matheus. The interestingness of deviations. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 25 – 36, Seattle, Washington, USA, July 1994. AAAI Press.

[132] John Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[133] Luc De Raedt and Albrecht Zimmermann. Constraint-based pattern set mining. In *Proceedings of the Seventh SIAM International Conference on Data Mining*, pages 237 – 248, Minneapolis, Minnesota, USA, April 2007. SIAM.

[134] Kimmo Raivio, Olli Simula, and Jaana Laiho. Neural analysis of mobile radio access network. In Nick Cercone, Tsau Young Lin, and Xindong Wu, editors, *Proceedings of IEEE International Conference on Data Mining*, pages 457 – 464, San Jose, California, USA, December 2001. IEEE Computer Society Press.

[135] Balász Rácz and András Lukács. High density compression of log files. In *Proceedings of 2004 Data Compression Conference (DCC 2004)*, page 557, Snowbird, Utah, USA, March 2004. IEEE Computer Society.

[136] Risto Ritala. Normative decision theory as a framework for operational decision support, Chapter 2. In Teemu Mätäsniemi, editor, *Operational decision making in the process industry, Multidisciplinary approach*, number 2442 in VTT research notes. VTT, Espoo, Finland, 2008.

[137] G. Rossi. A probabilistic model for measurement processes. *Measurement*, (34):85 – 99, 2003.

[138] Wolfgang Roth and Jürgen Baumann. Short message and data services; the general packet radio service (GPRS). In Friedhelm Hillebrand, editor, *GSM and UMTS. The Creation of Global Mobile Communication*, pages 425 – 430. John Wiley & Sons, Ltd, Chichester, England, 2002.

[139] Tobias Scheffer. Finding association rules that trade support optimally against confidence. In Luc De Raedt and Arno Siebes, editors, *Principles of Data Mining and Knowledge Discovery, 5th European Conference, PKDD 2001*, volume 2168 of *LNCS*, pages 424 – 435, Freiburg, Germany, September 2001. Springer.

[140] Jun Sese and Shinichi Morishita. Answering the most correlated N association rules efficiently. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002*, volume 2431 of *LNAI*, pages 410 – 422, Helsinki, Finland, August 2002. Springer.

[141] Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item sets that compress. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *Proceedings of the sixth SIAM International Conference on Data Mining*, pages 393 – 404, Bethesda, Maryland, USA, April 2006. SIAM.

[142] Herbert A. Simon. *The New Science of Management Decision.* Harper Brothers, New York, USA, 1960.

[143] Przemyslaw Skibinski and Jakub Swacha. Fast and efficient log file compression. In Yannis E. Ioannidis, Boris Novikov, and Boris Rachev, editors, *Communications of the Eleventh East-European Conference on Advances in Databases and Information Systems*, volume 325 of *CEUR Workshop Proceedings*, pages 56 – 69, Varna, Bulgaria, September 2007. CEUR-WS.org.

[144] Padhraic Smyth and Rodney M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301 – 316, August 1992.

[145] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 1(2):12 – 23, 2000.

[146] Roy Sterrit. Discovering rules for fault management. In *Proceedings of the eighth Annual IEEE International Conference and Workshop on Engineering of Computer Based Systems, ECBS 2001*, pages 190 – 196, Washington, DC, USA, April 2001. IEEE Computer Society.

[147] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction.* MIT Press, Cambridge, Massachusets, USA, 1998.

[148] Jari Suutarinen. *Performance Measurements of GSM Base Station System.* Lic.Thesis, Tampere University of Technology, 1994.

[149] TeleManagement Forum. Network management, detailed operations map. Evaluation version 1.0. GB908. TeleManagement Forum, March 1999.

[150] TeleManagement Forum. Telecom operations map. Approved version 2.1. GB910. TeleManagement Forum, March 2000.

[151] TeleManagement Forum. GB921 Business process framework release 8.0. TeleManagement Forum, November 2008.

[152] Joachim Tisal. *The GSM Network.* John Wiley Inc., Chichester, England, 2001.

[153] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes. Compression picks item sets that matter. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Proceedings of Knowledge Discovery in Databases: PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 4213 of *LNCS*, pages 585 – 592, Berlin, Germany, September 2006. Springer.

[154] Matthijs van Leeuwen, Jilles Vreeken, and Arno Siebes. Compression picks the significant item sets. Technical Report UU-CS-2006-050, Utrecht University, Department of Information and Computing Sciences, Utrecht, The Netherlands, 2006.

[155] Pekko Vehviläinen. *Data Mining for Managing Intrinsic Quality of Service in Digital Mobile Telecommunications Networks.* PhD thesis, Tampere University of Technology, P.O. Box 527, FIN-33101 Tampere, Finland, March 2004.

[156] Pekko Vehviläinen, Kimmo Hätönen, and Pekka Kumpulainen. Data mining in quality analysis of digital mobile telecommunications network. In *Proceedings of XVII IMEKO World Congress*, pages 684 – 689, Dubrovnik, Croatia, June 2003. IMEKO.

[157] Juha Vesanto. *Data Exploration Based on the Self-Organizing Map.* PhD thesis, Helsinki University of Technology, Laboratory of Computer and Information Science, P.O.Box 5400, FIN-02015 HUT, Espoo, Finland, September 2002.

[158] Jilles Vreeken, Matthijs van Leeuwen, and Arno Siebes. Characterising the difference. In Pavel Berkhin, Rich Caruana, and Xindong Wu, editors, *Proceedings of the thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, pages 765 – 774, San Jose, California, USA, August 2007. ACM.

[159] Jianyong Wang and George Karypis. SUMMARY: Efficiently summarizing transactions for clustering. Technical Report Technical report TR 04-025, University of Minnesota, Minneapolis, Minnesota, USA, 2004.

[160] Jianyong Wang and George Karypis. On efficiently summarizing categorical databases. *Knowledge and Information Systems*, 9(1):19 – 37, January 2006.

[161] Rüdiger Wirth and Jochen Hipp. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29 – 39, Manchester, United Kingdom, April 2000.

[162] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB 2005)*, pages 709 – 720, Trondheim, Norway, September 2005. ACM.

[163] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: A profile-based approach. In Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett, editors, *Proceedings of the eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 314 – 323, Chicago, Illinois, USA, August 2005. ACM.

[164] Mohammed Javeed Zaki. Generating non-redundant association rules. In *Proceedings SIGKDD'00*, pages 34 – 43, Boston, USA, August 2000. ACM Press.

[165] J. Ziv and A. Lempel. A universal algorithm for sequental data compression. *IEEE Transactions on Information Theory*, 22(1):337 – 343, 1977.

Appendix A

Requirements for knowledge discovery tasks

This Appendix presents characterisation of decision making levels of telecommunications network operations. The characterisations set requirements for the knowledge discovery process. The characterisations are linked to process phases that they affect most.

Table A.1: Summary of requirements for KD-tasks set by different types of decision situations.

	Automatic Control Functions	**Tactical Decision-making**	**Strategic Decision-making**	**Affected KD-process phase**
Supported tasks	Control circuits	Planning	Design	Knowledge requirements, UI, Form of results
	Error diagnostics	Configuration	Management support systems	
		Optimisation		
Actor	Operative automated systems	Expert systems, Human experts	Managers	UI, Form of results
Length of estimated time horizon	< n min	< n weeks	< n years	Knowledge requirements, Method selection, Analysis execution
Validity scope	Narrow, focused	Wider, focused to given sub goal	Wide, tied to scope of developed scenarios	Knowledge requirements, Method selection

Table A.1: Summary of requirements for KD-tasks set by different types of decision situations (continued).

	Automatic Conrol Functions	**Tactical Decision-making**	**Strategic Decision-making**	**Affected KD-process phase**
Type of decision	Objective, Subjectivity integrated to knowledge	Objective / Subjective towards risk, Multi-objective	Multi-objective, Subjective	Knowledge requirements
Mode of support	Off-line knowledge extraction and executable model construction	Off-line knowledge extraction and human interpretable model construcation	Off-line knowledge extraction and human interpretable model construcation	UI, Form of results, Method selection, Analysis execution
	On-line adaptivity	On-line support for decision tasks		
KD-iteration time	Days – weeks	In on-line support minutes – hours	In on-line support: days	Method selection, Analysis execution
		In off-line support days – weeks	In off-line tasks: > weeks	
Decision tasks	System state identification, System state estimation	System state identification System state estimation	Cost estimation System state estimation External estimation Cost estimation	Knowledge requirements
What causes information need	Changes in system inputs and service requests	Changes in system inputs, service requests and direct surroundings	Changes in external factors	Knowledge requirements
	Changes in system surroundings	Changes in system policies	System upsets	
	Changes in system policies	Changes in system parameters	Business upsets	

Table A.1: Summary of requirements for KD-tasks set by different types of decision situations (continued).

	Automatic Conrol Functions	**Tactical Decision-making**	**Strategic Decision-making**	**Affected KD-process phase**
	Changes in system parameters	Changes in system production portfolio		
	Changes in product portfolio	System upsets		
Knowledge requirement with respect to the phase of the system lifespan	Frequent at the beginning of the system lifespan	Frequent throughout the system lifespan	Depends on the business	Method selection, Knowledge require-ments, Analysis execution
	Later validation and updating when system parameters, product portfolio, or policies are changed		Frequent follow-up	
Type of available data	Measurements as time series	Measurements and cost data as time series	Aggregated measurements, benchmarking, statistics	Preprocessing, Method selection, UI, Form of results
	Event logs	Event logs	Free-text reports	
		Semi-structured documents	Behavior and business models and theories	
Required content	Dynamic system models System state descriptions	Plans, scenarios System state descriptions	Scenarios	Knowledge requirements

Table A.1: Summary of requirements for KD-tasks set by different types of decision situations (continued).

	Automatic Conrol Functions	**Tactical Decision-making**	**Strategic Decision-making**	**Affected KD-process phase**
Required type of KD task results	Human verifiable mathematical or logical model	Models understandable for humans	Any human understandable knowledge	Method selection, UI, Form of results
Information quality	Robustness	Generalisability	Coherency	Postprocessing and Validation
			Coverage	
Required validations	Response testing	Comparison against history	Comparison against history	Postprocessing and Validation
		Simulations	Large-scale simulations	

Appendix B

QLC Queries

This Appendix lists the queries used in testing query evaluation against QLC compressed files. Table B.1 lists queries applied to the small data log files and Table B.2 lists queries applied to the large data log files. The queries have been anonymised without changing field names or query structure.

Table B.1: Queries to daily log files of small data with answer size minimum, median and maximum. The queried values have been anonymised.

Class	No	Query	Min	Median	Max
A	0	User:smith	0	0	0
A	1	Action:xxxxxx	0	1	11
A	2	Service:xxxx	0	0	4
A	3	Source:xxxx	0	22	252
A	4	Protocol:tcp && S_port:xxxx	0	0	1
A	5	Service:xxxx && Destination:123.123.123.123	0	0	878
B	6	Origin:foo.bar.xxx	3411	5132	15587
C	7	Service:http	45	238	962
C	8	Service:http && Info:xxxxxx	38	50	325
C	9	Service:http && S_port:x[0-9]+	0	21	107
C	10	Service:http && Destination:123.+	0	0	24
C	11	Service:http && S_port:x[0-9]+ && Info:xxxxxx	0	13	101
C	12	Service:http && S_port:x[0-9]+ && Destination:123.+	0	0	0
C	13	Service:http && S_port:x[0-9]+ && Destination:123.+ && Info:xxxxxx	0	0	0
D	14	Destination:123.123.123.123	413	681	682
D	15	Destination:123.123.123.123 && Info:xxxxxx	413	681	682
D	16	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx	413	681	682
D	17	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx	413	681	682
D	18	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx && Rule:xx	413	681	682
D	19	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx && Rule:xx && S_port:xxxx	413	681	682
D	20	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx && Rule:xx && S_port:xxxx && Service:xxxx	413	681	682

Table B.1: Queries to small data (continued).

Class	No	Query	Min	Median	Max
D	21	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx && Rule:xx && S_port:xxxx && Service:xxxx && Source:xxxx	413	681	682
D	22	Destination:123.123.123.123 && Info:xxxxxx && Interface:xxxx && Protocol:xxxx && Rule:xx && S_port:xxxxxx && Service:xxxx && Source:xxxx && Type:log	413	681	682
E	23	Destination:123.123.123.123	1651	2111	2806
E	24	Destination:123.123.123.123 && Interface:xxxx	1651	2111	2806
E	25	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp	1651	2111	2806
E	26	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp && Rule:xx	1651	2111	2806
E	27	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp && Rule:xx && S_port:xxxx	1651	2111	2806
E	28	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp && Rule:xx && S_port:xxxx && Service:xxxxxx	374	480	1194
E	29	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp && Rule:xx && S_port:xxxx && Service:xxxxxx && Source:xxxx	0	96	102
E	30	Destination:123.123.123.123 && Interface:xxxx && Protocol:udp && Rule:xx && S_port:xxxx && Service:xxxxxx && Source:xxxx && Type:log	0	96	102
F	31	Protocol:udp	2688	3565	4323
F	32	Protocol:udp && Action:xxxxxx	2688	3564	4322
F	33	Protocol:udp && Action:xxxxxx && Destination:123.123.123.123	1651	2111	2806
F	34	Protocol:udp && Action:xxxxxx && Destination:123.123.123.123 && Interface:xxxx	1651	2111	2806
F	35	Protocol:udp && Action:xxxxxx && Destination:123.123.123.123 && Interface:xxxx && Service:xxx.+	1627	2087	2782
F	36	Protocol:udp && Action:xxxxxx && Destination:123.123.123.123 && Interface:xxxx && Service:xxx.+ && Source:xxxx	0	204	216

Table B.2: Queries to daily log files of large data with answer size minimum, median and maximum. The queried values have been anonymised.

Class	**No**	**Query**	**Min**	**Median**	**Max**
A	1	reason:xxxxxxxxxxxxxxxxxxxxxx,	10	68	168
A	2	reason:yyyyyyyyyyyyyyyyyyyyyy,	8	81	490
A	3	reason:.+zzzzzzz,	19	135	437
A	4	user:asmith,	0	12	590
A	5	user:bsmith,	26	188	5092
A	6	action:xxxxxx,	0	14	55
B	7	alert:xxxx,	77674	595623	1119850
B	8	action:yyyyyy,	8471	12648	19569
B	9	i/f_dir:xxxxxx && proto:xxxx && type:log,	4428	5306	6440
B	10	action:zzzzzz && i/f_dir:xxxxxxx,	1512	1976	3325
C	11	action:vvvvvv && i/f_dir:xxxxxxx,	0	0	0
D	12	user:[a-zA-Z0-9]+,	53	499	5336
D	13	action:zzzzzz && user:[a-zA-Z0-9]+,	29	215	747
E	14	action:zzzzzz,	23761	56941	128769
E	15	action:zzzzzz && i/f_dir:xxxxxx,	21803	54420	126379
E	16	action:zzzzzz && i/f_dir:xxxxxx && i/f_name:xxxxxx,	20377	52669	124944
E	17	action:zzzzzz && i/f_dir:xxxxxx && i/f_name:xxxxxx && proto:tcp,	20377	52663	124936
E	18	action:zzzzzz && i/f_dir:xxxxxx && i/f_name:xxxxxx && proto:tcp && rule:xx && type:log,	20351	52403	124648
F	19	action:zzzzzz,	23761	56941	128769
F	20	action:zzzzzz && proto:xxxx,	1907	2334	3081

Table B.2: Queries to large data (continued).

Class	No	Query	Min	Median	Max
F	21	action:zzzzzz && proto:xxxx && rule:xx && type:log,	1300	1730	2187
F	22	action:zzzzzz && len:xx && proto:xxxx && rule:xx && type:log,	0	0	0
G	23	action:uuuuuu,	295	890	32388
G	24	action:uuuuuu && i/f_dir:xxxxxx,	295	890	32388
G	25	action:uuuuuu && i/f_dir:xxxxxx && type:xxxxxx,	295	890	32388
H	26	action:vvvvvv,	38557	501677	1038208
H	27	action:vvvvvv && i/f_dir:xxxxxx && proto:udp,	26681	473859	1012414
H	28	action:vvvvvv && i/f_dir:xxxxxx && service:xxxx,	667	436946	977132
H	29	action:vvvvvv && i/f_dir:xxxxxx && s_port:xxx[0-9],	283	418095	973656
H	30	action:vvvvvv && i/f_dir:xxxxxx && s_port:xxx0,	11	558	330405
I	31	action:vvvvvv && i/f_dir:xxxxxx && i/f_name:if1,	24836	47638	212003
I	32	action:vvvvvv && i/f_dir:xxxxxx && i/f_name:if1 && proto:udp,	13771	28598	184977
I	33	action:vvvvvv && i/f_dir:xxxxxx && i/f_name:if1 && proto:udp && service:xxxx,	63	7945	163350
I	34	action:vvvvvv && i/f_dir:xxxxxx && i/f_name:if1 && proto:udp && service:xxxx && s_port:xxxx	0	129	764