NTNU
Innovation and Creativity

A Survey of Combining Association Rules for Pre-warning of Oil Production Problems

Per Kristian Helland

Master of Science in Computer Science
Submission date: May 2007
Supervisor: Torulf Mollestad, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

This project aims to explore the problem of how to reveal future problems in a production process, produce appropriate alarms and thereby allowing efficient handling
of an unwanted and potentially dangerous situation. Data mining techniques will be applied to uncover event patterns in time series generated from sensors monitoring the oil production, and prototypes for utilization of such patterns will be implemented.

Assignment given: 16. January 2007
Supervisor: Torulf Mollestad, IDI

Abstract

Periods of sub-optimal production rates, or complete shut-downs, add negative numbers to the revenue graph for oil companies. Oil and gas are produced from several reservoirs and through many wells with varying gas/oil proportion, making it a complex process that is difficult to control. As a part of a three-step process for utilizing data in the oil production domain, this thesis derive methods for combining event patterns, called restricted association rules, in time series in order to warn about future anomalies in oil production processes. Two problems have been considered: Network learning and network reasoning. The suggested solution consists of building an Association Rules Network (ARN) from the rule set given as input. After transforming the hypergraph-based ARN to a directed acyclic graph, correlations between nodes are found by applying the shortest-path principle. Motivated by the shortcomings of this simple solution, it is shown how a method for learning Bayesian networks with support for representation of temporal dependencies can be derived from the initial ARN. The concept, named Temporal Bayesian Network of Events (TBNE), is a powerful, but yet complex solution that enjoys the properties of Bayesian network reasoning while at the same time representing temporal information. This thesis has shown that it is theoretically feasible to combine restricted association rules in order to create a network structure for reasoning. It is concluded that the final choice of solution must be based on a carefully consideration of the trade-off between complexity and expressiveness, and that a natural continuation is testing the suggested concepts with real data.

Preface

"Knowledge is power." — Francis Bacon (1561–1626)

The challenges I have faced during the work of this master thesis have been both exciting and frustrating. Luckily, the satisfaction of a good idea beats the frustration of being stuck with a difficult problem for three days.

I have written this thesis as a fifth year student of the Master studies in Computer Science at the Norwegian University of Science and Technology. My background is mainly from the field of machine learning and AI, but statistics is also a field of interest. As a part of a two-man army working on the same problem case at ConocoPhillips, I would like to thank Joacim Christiansen for his help and support. Our discussions have been crucial to the final results.

Torulf Mollestad, my supervisor, has shared his expertise and experience, but also challenged me to think and re-think about my thoughts and suggestions. His support and advice has been invaluable to me along the way.

Trondheim, 31/05/2007

Per Kristian Helland

Contents

1 Introduction **7**
- 1.1 Research goals . . . 7
  - 1.1.1 Problem description . . . 8
  - 1.1.2 Limitations of scope . . . 9
- 1.2 Background and motivation . . . 9
- 1.3 Data to Decision: The greater picture . . . 10
  - 1.3.1 Remote decision making . . . 10
  - 1.3.2 Problem case . . . 11
  - 1.3.3 Guidelines for data utilization . . . 12
- 1.4 Report outline . . . 15

2 Preliminaries **16**
- 2.1 Probability theory . . . 16
- 2.2 Expert systems . . . 18
  - 2.2.1 Rule-based systems . . . 19
  - 2.2.2 Probabilistic expert systems . . . 19
- 2.3 Association rule mining . . . 21
- 2.4 Hypergraph theory . . . 23

3 Learning network structures **25**
- 3.1 Assumptions on learning input . . . 25
  - 3.1.1 Rule syntax and semantics . . . 26
  - 3.1.2 Rule cleaning . . . 28
- 3.2 Association rules network . . . 28
  - 3.2.1 Learning ARN . . . 29
  - 3.2.2 Improving the ARN algorithm . . . 31
- 3.3 Bayesian networks . . . 33
  - 3.3.1 Learning Bayesian networks . . . 33
  - 3.3.2 Time modelling in Bayesian networks . . . 35
  - 3.3.3 Issues by combining Bayesian networks and association rules . . . 36

4 Reasoning in network structures **39**
- 4.1 Reasoning in ARN . . . 40
  - 4.1.1 Fundamentals of ARN reasoning . . . 40
  - 4.1.2 Transforming an ARN to a DAG . . . 41
  - 4.1.3 Explaining correlation between nodes . . . 42
  - 4.1.4 Reasoning with temporal information in ARN . . . 44
- 4.2 Reasoning in Bayesian networks . . . 45

4.2.1 Fundamentals of Bayesian network reasoning 45
4.2.2 Reasoning with temporal information in Bayesian networks 47
4.3 Comparing the suggested reasoning schemes 48

5 Discussion and conclusion **52**
5.1 Suggested solutions 52
5.1.1 A simple solution 52
5.1.2 An advanced solution 54
5.2 Evaluating the overall solution 55
5.3 Contributions 57
5.4 Conclusion 57

Bibliography **59**

A Data and calculations **63**
A.1 Marginalization example of Chapter 2.2.2 63
A.2 Data for the example of Chapter 4.3 64

List of Tables

1.1 Step 1: Data preprocessing . . . 13
1.2 Step 2: Discovering dependencies between events . . . 14
1.3 Step 3: Knowledge utilization . . . 15

2.1 Conditional probability table for having a headache . . . 20
2.2 Conditional probability table for having fever . . . 20

3.1 Possible rule structures . . . 26
3.2 Possible ordering of events . . . 27
3.3 Assumptions made on rule syntax and semantics . . . 28
3.4 CPT for the DLH node . . . 36

4.1 Explanation of transformed hyperedge weights . . . 42
4.2 Reasoning with temporal information in ARNs. . . . 45
4.3 Comparing the reasoning mechanisms of ARN and TBNE . . . 49

5.1 Properties of the suggested solutions . . . 55

A.1 Calculation of the marginal probability $Pr(Headache = yes)$ . . . 64
A.2 Node A . . . 64
A.3 Node B . . . 64
A.4 Node C . . . 65
A.5 Node D . . . 65
A.6 Node E . . . 65
A.7 Node F . . . 65
A.8 Node X . . . 65

List of Figures

1.1 Example illustrating sub-optimal production . . . 8
1.2 Integrated operations . . . 11
1.3 Production process 2/4 J . . . 11
1.4 A simplified low pressure separator . . . 12
1.5 Illustration of the overall three-step data mining vision. . . . 13

2.1 Bayesian network for the headache domain . . . 20
2.2 Illustrations of probability evolution for the headache example . . . 21
2.3 Example of B-arc and F-arc . . . 23
2.4 B-graphs . . . 23

3.1 Definition of temporal relations using the rule $X_1, X_2 \rightarrow Y$ . . . 27
3.2 ARN with reverse hyperedge . . . 30
3.3 Bayesian network example . . . 33
3.4 Bayesian network learning . . . 34
3.5 A TBNE example . . . 36

4.1 Illustration of hypergraph transformations. . . . 41
4.2 Illustrations of the weight functions. . . . 43
4.3 Illustration of the triangulation process . . . 46
4.4 The join tree and junction tree for the triangulated graph in Figure 4.3c . . . 46
4.5 Generic network learned from a fictitious rule set . . . 49
4.6 TBNE and ARN after transformation . . . 50
4.7 TBNE after the occurrence of A and E respectively . . . 50

Chapter 1

Introduction

For oil companies, the combination of today's record-high oil prices and stakeholders strict demands for return on investments, combined with sparse oil and gas resources, brings forth the need for business optimization. Onshore operation centres (OOC) are extensively supporting offshore operations using state of the art communication technology. This reduces the need for offshore personnel and is a cost effective way of managing operations. It also introduces the possibility for additional monitoring systems that can be operated onshore.

Oil production is a non-trivial process and it differs in nature from reservoir to reservoir. More sophisticated technology makes it possible to do drilling operations that would have been impossible only few years ago, like horizontal drilling. The new possibilities also increase the complexity of oil production since e.g. oil and gas are produced from several reservoirs and through many wells with varying gas/oil proportion.

In oil production periods of unplanned sub-optimal production rates, or complete shut-downs, add negative numbers to the revenue graph. For instance, ConocoPhillips loses 1.4 million Danish Kroner (about 175 000 Euro) in income for each hour an oil well is not in production [21]. On average, ConocoPhillips calculates with around 40 unplanned well shut downs per year, and the total costs become severe. A system that could give an early warning about possible future anomalies in the oil production, and do this in time for the operators to react, would therefore be valuable in order to reduce costs.

1.1 Research goals

Monitoring oil production processes is done in order to ensure that they are effective. Such monitoring usually involves several sensors, which are used to depict a variety of attributes of the process over time. Example of attributes are temperature, pressure and valve openings, among others. Measurements are done on those parts of the system which are considered critical for the operation and process stability. The choice of attributes to monitor is critical, and are mainly performed by domain experts. The frequency of sampling is chosen by the user and may vary from a few milliseconds to many days dependent on the attribute in question. The observation data, and measurement of production rates, are stored in time series databases.

For ConocoPhillips elimination of production loss is an important goal, and research on ways of predicting production stops and loss of quality in production is an area of commitment. Both complete shut-downs and sub-optimal production rates result in a considerable loss of money. An example of a possible production rate time series is given in Figure 1.1. It illustrates a characteristic full stop in production and an interval of sub-optimal production (shaded area).

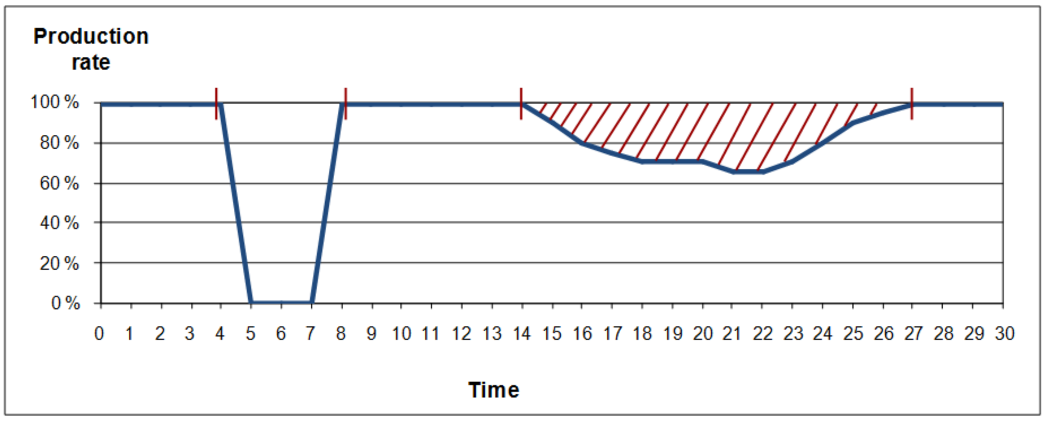


Figure 1.1: Example illustrating sub-optimal production

In general, methods for searching large volumes of data for patterns belongs to the field of data mining. One goal is to characterise the general properties of a data collection and/or perform inference on the current data in order to make predictions. The monitoring of sensors in oil production produces a large amount of data, providing a breeding ground for a data mining hypothesis that patterns can be found in the historical data, and subsequently be applied to real time production data in order to warn about future production rate anomalies.

1.1.1 Problem description

ConocoPhillips wants to explore the problem of how to reveal future problems in a production process, produce appropriate alarms and thereby allowing efficient handling of an unwanted and potentially dangerous situation. This particular master thesis aims to investigate how a set of uncovered event patterns in time series, called restricted association rules, can be utilized. More specific, the goal is to reach a conclusion of the feasibility of combining restricted association rules in order to warn about future production problems.

The problem is divided into two sub-problems:

1. **Network learning:** How to learn a network structure from restricted association rules that makes it possible to define some kind of reasoning mechanism to decide whether an alarm should be given or not.
2. **Network reasoning:** How to define the reasoning mechanism for failure prediction in the learned network. An alarm system will have to deal with uncertain information, and the reasoning must define how the system's belief of a future production anomaly should be stated.

In a broader perspective the thesis contributes to the field of expert systems, i.e. the problem of codifying and utilizing knowledge obtained from experts or other sources. The thesis should have a theoretical approach to the problem as real production process data will not be available.

1.1.2 Limitations of scope

This master thesis' main goal is to reach a conclusion of the feasibility of combining restricted association rules in order to warn about future production problems. It is, however, not an implementation task, and challenges of integrating suggested solutions with existing systems in ConocoPhillips will neither be considered.

It is also important to note the difference between predicting the production rate and predicting possible production anomalies. Predicting production rate may be seen as predicting a graph based only on the history of this single graph, analogue to predicting a stock price. Predicting possible production anomalies is a broader problem concerning prediction of variables contributing to a higher level production anomaly. This work contributes to the latter, with the ultimate goal of giving early warnings of faults in order to make them avoidable.

1.2 Background and motivation

ConocoPhillips was founded in 2002 after the merging of Conoco Inc. and Phillips Petroleum Company. It is among the largest non-government controlled energy companies worldwide, having, at year-end 2005, 38 400 employees and assets of $165 billion USD [3]. ConocoPhillips' core activities worldwide are:

- Petroleum exploration and production.
- Petroleum refining, marketing, supply and transportation.
- Natural gas gathering, processing and marketing.
- Chemicals and plastics production and distribution.

The oil industry is one of the largest and most profitable industries in the world. It is also one of the most cost aware industries. Since the price of oil is standardized, oil companies concentrate on cutting internal production costs and optimizing production.

ConocoPhillips Norge accounts for about 14 percent of the oil and gas production in ConocoPhillips worldwide. It has a total of 1735 employees and is the third-largest energy company in Norway [21]. In January 2005, ConocoPhillips started a program called From Good to Great which directly involves over 700 employees. The goal of From Good to Great is to improve processes and increase efficiencies to cut costs and optimize production. The hierarchy of these goals are:

1. Maximize production and increase the process regularity.
2. Reduce costs.

Fault detection is a key issue in the development of complex oil production facilities. Several techniques, including alarms on sensors and process simulation, have been tried to avoid failures or loss in production with varying degree of success. Monitoring based on process knowledge is mainly done by domain experts, but as the complexity of these systems grow this becomes a difficult task to manage. Such monitoring requires a tremendous amount of a priori knowledge of the system behaviour for each facility, whereas a large part of this knowledge is often tacit and not easily obtained. For example deriving a complete set of production rules for a facility is usually a time consuming task, and too expensive to do for each facility. In addition, re-use of knowledge from other facilities is also limited, because each facility is usually more or less different from others. Another aspect is introducing new operators, which in general is a huge challenge because of the tacit and not easily obtained knowledge. Therefore another goal is to make this knowledge easily available to new operators.

Prediction problems has received great attention in the statistical and financial domain, e.g. attempts to predict stock prices, but results have been relatively bad with respect to correctness of predictions. While the stock market is a non-deterministic process with a huge amount of uncertainty, the oil production process can be assumed to be deterministic and describable by a finite set of states. Oil production is limited by physical laws and tend to operate in a closed environment. This and the hypotheses that there exists a logical and statistical correlation between sensor data and production anomalies that may be found, motivates for a possible data mining solution.

1.3 Data to Decision: The greater picture

One pillar of From Good to Great is Data to Decision (D2D). D2D directly involves around 40 employees and is the IT infrastructure to support From Good to Great. It is applicable in one oil field in the North Sea named Ekofisk. This field represents approximately 10% of the oil company's worldwide oil production. The daily field production is an estimated 375 000 barrels of oil, equaling around 22 million USD in daily revenues [21]. D2D deals with the problem of how the vast number of stored data related to oil production can be used to make faster and better decisions.

1.3.1 Remote decision making

An ultimate goal for data utilization in the context of predicting production anomalies is to create a failure prediction system. A likely scenario for a system of that kind is that it will be integrated in an OOC. An OOC is a digital operation centre located onshore which is connected to offshore drilling and production facilities by fibre wires. Figure 1.2 places OOCs in the greater picture of what is called Integrated Operations (IO), the future of oil production where technology that gathers competence, data and applications in real-time, independent of distance, is extensively utilized. Information sent from operational instruments makes it possible to monitor and control the platforms remotely. Daily meetings between the operational management located onshore and offshore has brought the two environments closer. In addition, real-time data is distributed to collaborators, like maintenance actors. A focus area for the OOC of ConocoPhillips Norway is production optimization, and elimination of production loss is an important goal.

The technology available at an OOC makes it possible to operate a failure prediction system there, and the vision is to build a system that receives real-time sensor data from oil production as input, and con-

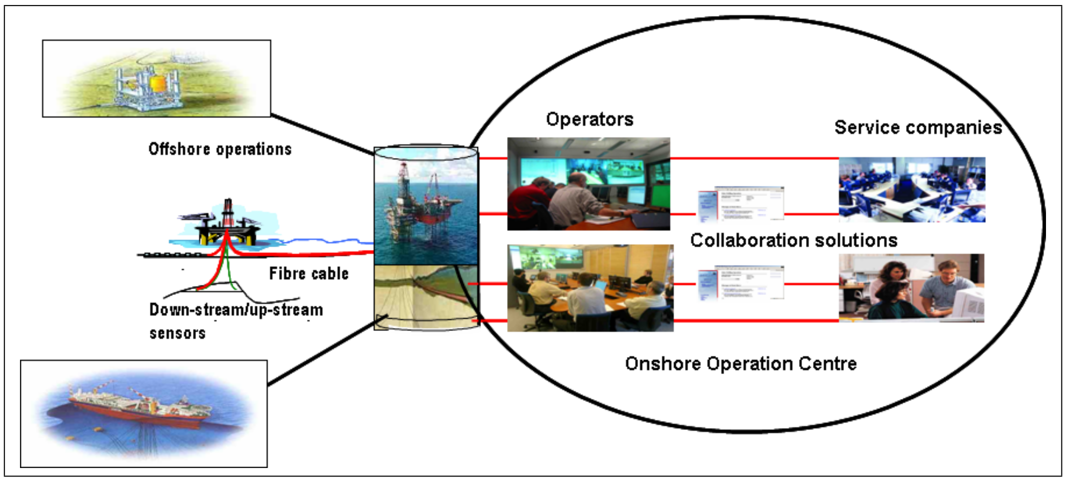


Figure 1.2: Integrated operations

tinuously processes the data in order to discover important events and reason about possible failures. Ideally the production state should be visualised on a large screen, showing events and their dependencies. If a warning is given, the system should explain the cause and suggest corrective actions. Such a system will probably reduce the amount of tacit knowledge needed when monitoring oil production because this knowledge is communicated through the visualised production state. The ability to predict more complex situations early will probably be increased since the prediction, which might include a great amount of data, is handled by computers.

1.3.2 Problem case

The specific production process that has been used as a test case in D2D is named "2/4 J" and is sketched in Figure 1.3.

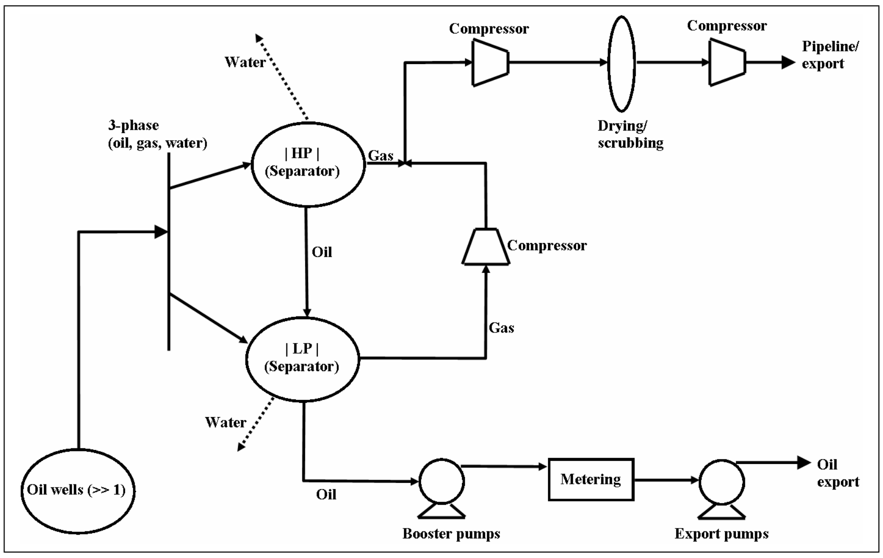


Figure 1.3: Production process 2/4 J

This simplified figure shows how 3-phase flows of gas, oil and water are lead into a high pressure or a low pressure separator. The water is cleaned and then discharged into the sea, while oil and gas is sent to onshore refineries. Throughout this process sensors[1] are monitoring various attributes, as mentioned earlier.

An emulsion is a mixture of two immiscible substances, and for a 3-phase separator this can happen for oil/water. The problem case targeted is the prediction of emulsification in the low pressure separator of the 2/4 J process. A sketch of the separator is shown in Figure 1.4.

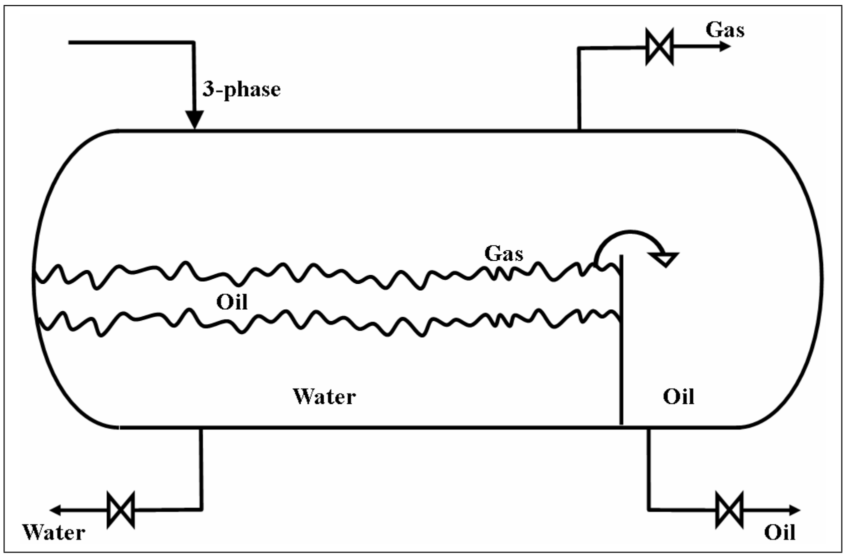


Figure 1.4: A simplified low pressure separator

As shown, oil and water leave the vessel at the bottom through different valves, and the gas leaves the vessel at the top. If the water level is to high, both water and oil will flow over the edge. A high amount of water gives low quality oil, and hence a lower price. If emulsification occurs, the water level is impossible to measure, and corrective actions have to be done.

1.3.3 Guidelines for data utilization

This study is related to D2D and in particular to the work done by Senior Consultant at SAS Institute, and Associate Professor II at NTNU, Torulf Mollestad [40, 41]. Mollestad has developed a general guideline for data utilization in the oil production domain, and under his supervision a project thesis investigating this guideline was written by Christiansen and Helland [18] autumn 2006. Figure 1.5 shows the suggested three-step process, going from sensor data to an operational system.

[1]The terms "sensor" and "tag" are used alternating.

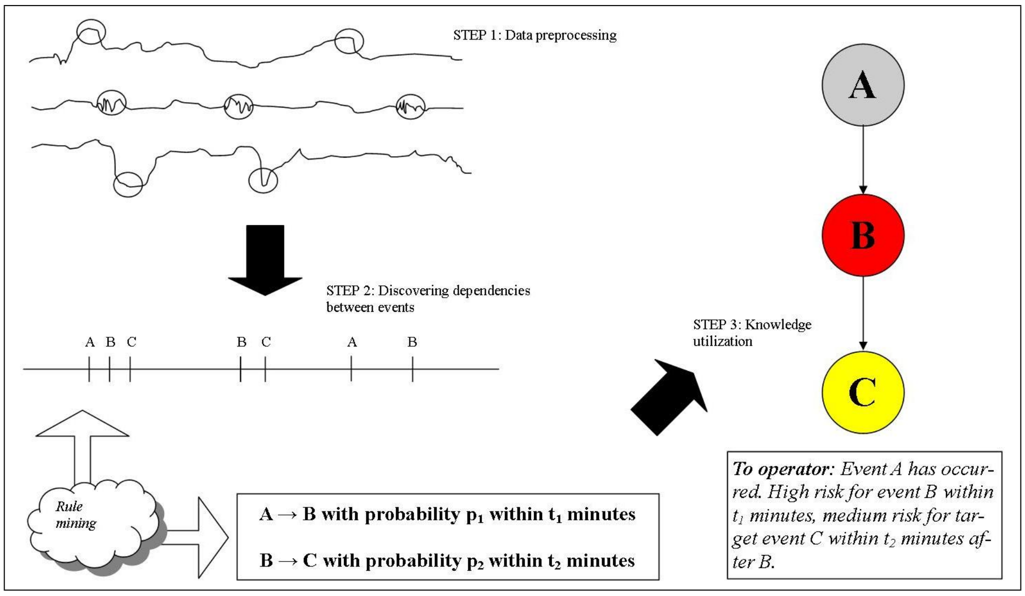

STEP 1: Data preprocessing
A
B
C
STEP 2: Discovering dependencies between events
STEP 3: Knowledge utilization
A B C B C A B
Rule mining
A → B with probability p_1 within t_1 minutes
B → C with probability p_2 within t_2 minutes
To operator: Event A has occurred. High risk for event B within t_1 minutes, medium risk for target event C within t_2 minutes after B.

Figure 1.5: Illustration of the overall three-step data mining vision.

The process illustrated by Figure 1.5 is not trivial, and each step is next explained in more detail.

1. Data preprocessing includes data loading, data cleaning, event discovery and variable clustering, as explained in Table 1.1.

Task	**Description**
Data loading	The process of loading sensor data from the database where it is located. The amount of available data is in many cases greater than what is manageable for further processing. A decision of what sampling rate to use must be taken.
Data cleaning	The process of filling in missing values, smoothing out noise and correction of data inconsistency.
Event detection	The process of discovering unusual and/or important events in the time series. An event must be explainable to a user in order to be useful. Basic techniques as threshold measures are often suitable because they are computational efficient and easy to interpret.
Variable clustering	The process of removing variables (event series) that do not influence on production loss. Removing such uncorrelated variables, which also may be seen as noise, is beneficial because it may reduce running times and improve the results of association rule mining algorithms.

Table 1.1: Step 1: Data preprocessing

For the particular problem case of emulsification in the low pressure separator of process 2/4 J, Mollestad has developed a framework for finding relevant properties of sensor data. The framework, that conforms to the step of data preprocessing, can be divided into two main tasks:

- *Data transformation:* Let domain experts select tags (sensors) that are considered potentially relevant. Apply transformation functions to each time series (data from selected tags) in order to derive binary event time series. Examples of transformation functions are: Extremely high/low levels, high volatility in data, long/dramatic shifts in the data, passing of (predefined) critical levels, and user defined functions. In order to continue to the next step, a target event must be defined by domain experts.

- *Variable clustering:* Cluster the binary event time series in order to identify events that co-occur with the target event. The goal of this process is to remove uninteresting and/or non-significant variables, and the resulting events are those related to problem situations, i.e. they are potential warning signals.

For the 2/4 J process, 119 tags were selected as potentially relevant, and the transformation process generated 1050 event time series. The target event defined by experts was identified to happen 22 times in the data from December 2005 to December 2006. The variable clustering then reduced the set of binary event time series to about 200.

2. Discovering dependencies between events (rule mining) includes the concepts of sliding window extraction, event sequence mining, restricted association rules, interestingness measures and rule cleaning, as explained in Table 1.2.

Task	**Description**
Sliding window extraction	The process of splitting a sequence of different events into groups of events occurring within some fixed time interval.
Restricted association rules	Association rules which describes association between occurrences of events or combination of events, taking temporal relations into account. Such rules may be found in the historical data and used to predict future events.
Interestingness measures	Primarily mathematical expressions that represent the interestingness of rules. Such expressions may be used to efficiently prune the search space during rule mining, but also in rule cleaning/selection.
Rule cleaning	The process of identifying interesting rules and removing non-interesting, redundant or unwanted rules. Such a process may be done automatically or by a human expert using an interactive rule cleaning tool.

Table 1.2: Step 2: Discovering dependencies between events

Parallel to this thesis work on the second step has been done by Christiansen [17]. The main findings of Christiansen's work is that association rules suited for prediction differs from regular association rules. By restricting the rule mining task, rules suited for prediction can be targeted and efficiently discovered from time series. The number of rules found is usually vast, but by removing redundant rules and ranking them with respect to interestingness and information content, a small subset of rules can be obtained. Association rules found in this specific way are theoretically well suited for application in reasoning structures, and are believed to give good predictive performance.

3. Knowledge utilization is the final step and involves learning of a network structure, definition of network reasoning and operational use, and the concepts are further explained in Table 1.3.

Part one and two of the third step conform with the problem description of Chapter 1.1.1 and are the focus areas of this thesis.

Task	**Description**
Network learning	The task of building a network based on the discovered restricted association rules and their dependencies.
Network reasoning	The process of drawing inferences appropriate to the situation, i.e. update the belief on events in order to predict failures.
Operational use	Consists of to parts: (1) How to make suggestions of corrective actions (technical), and (2) how to integrate the system with other monitoring systems (system integration).

Table 1.3: Step 3: Knowledge utilization

1.4 Report outline

The remainder of this report is structured as follows. Chapter 2 introduces important concepts related to the problem to be discussed. The main goal of this chapter is to form a theoretical foundation for the following discussions. In Chapter 3 a method for learning a network structure from a set of restricted association rules is derived. The method relates to the concept of Association Rules Network (ARN). Due to shortcomings of the ARN concept, Bayesian networks are introduced as an alternative. Chapter 4 gives a detailed study of how to reason in the networks introduced, and compares them by analysing an example. Finally, Chapter 5 considers a short-term solution that can be implemented and tested immediately at ConocoPhillips, followed by a discussion taking a long-term view on the issue. A discussion is also given considering the overall process of network learning and reasoning, as well as the three-step process of data utilization in the oil production domain. The chapter also discusses this master thesis' contributions to the field of expert systems before it ends with a conclusion.

Chapter 2

Preliminaries

This chapter describes the concepts of probability theory, expert systems, association rules and hypergraphs. Understanding *probability theory* is fundamental in order to discuss the effect of combining rules in a reasoning network. When all is said and done, the goal is to express the degree of belief the system has on a production failure, hence having an understanding of probability is important. *Expert systems* codify and utilize knowledge obtained from experts or other sources. A brief introduction is given so that the reader unfamiliar with the concept will have an understanding of the field of research. In the preceding step of the suggested process described in Chapter 1.3.3 the concept of *association rule mining* is crucial. Restricted association rules, as they will be named here, are given as input to the learning process, so basic terminology is introduced. Finally, one of the main suggestions of how to learn a reasoning network from a rule set is built upon the notion of *hypergraphs*. Since this concept is not very common, a section describing basic hypergraph theory ends this chapter.

As the chapter name indicates, the intention is to introduce underlying theory necessary to understand what follows. Readers familiar with some or all of these concepts may skip the relevant parts.

2.1 Probability theory

Incomplete and uncertain information is unavoidable in practical artificial intelligence (AI) systems[1] and must be dealt with in a careful matter. A broad range of possible solutions have been suggested over the past decades. In a review of the subject, Parsons and Hunter [46] split uncertainty formalisms in two main directions; numerical[2] and symbolic methods. The overall solution described in Chapter 1.3.3 clearly emphasizes a use of the former method.

Numerical methods include probability theory, evidence theory (also called Dempster-Shafer theory) and possibility theory. These methods are, among others, evaluated in the light of expert systems by Ng and Abramson [44]. Uncertainty is connected to the amount of belief one has of one or more events to occur. How this belief should be assigned is a contentious issue, but traditionally the belief assigned is a number between 0 and 1, where 0 is assigned to facts known to be false and 1 is assigned to facts known to be true. Some theories limits the total belief that may be assigned in such way that the sum of all

[1]Expert systems belongs to the class of AI systems.
[2]The term *quantitative methods* is also in use.

the beliefs becomes 1, but this is not a constraint in e.g. possibility theory. Similarly, probability theory limits the belief of a hypothesis based on the belief assigned to its negation. As pointed out by [46], an essential feature of artificial intelligence applications is to combine beliefs that results from several different information sources. This is also something that the theories threat in different ways.

The term "probability" can be interpreted in several ways, none of which can claim to be the only correct interpretation. Three views have dominated since the turn of the century [44]:

- **Objectivistic:** Probability is a measure for the limit of an event's relative frequency in a large number of trials. In other words, it is guaranteed by the law of large numbers [22] that the percentage of an event's occurrences will approach an objective probability given enough observations. Another term used is "frequency probability".
- **Subjectivistic:** Probability is a measure of a person's belief in a certain proposition. With this view, it is possible that two person faced with the same evidence can have different degrees of confidence in a given proposition's truth. Other terms for subjective probability are "Bayesian" and "personal".
- **Logical:** Probability is a measure for inferential soundness. It measures, out of logical necessity, the extent to which one set of propositions confirms the truth of another. Logical probability is also referred to as "necessary probability".

As written by [46], belief may be distributed on the basis of statistical information, physical possibility, or purely subjective assessment by an expert, to mention some. Also, given the result of a belief distribution, what is interesting is how the assigned beliefs may be manipulated. These questions are the core of Chapter 3 and 4 respectively.

Lindley [36] claims that probability theory is based on three laws that define the behaviour of probability measures:

1. *The convexity law* states that the probability measure for an event A given information H is such that:
$$0 \leq Pr(A|H) \leq 1$$
2. *The addition law* refers to the probability of the union of two events. For two exclusive events (that is, they cannot both occur) this becomes:
$$Pr(A \cup B|H) = Pr(A|H) + Pr(B|H)$$
or just
$$Pr(A \cup B) = Pr(A) + Pr(B)$$
If the two events are not exclusive, the calculation becomes:
$$Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$$
Furthermore, for a set of n mutually exclusive and exhaustive events A_i, the sum of the probabilities is equal to 1:
$$\sum_{i=1,\ldots,n} Pr(A_i) = 1$$

3. *The multiplication law* gives the probability of two events A and B occurring together, that is, the probability of the intersection of A and B:

   $$Pr(A \cap B|H) = Pr(A|H) \cdot Pr(B|A \cap H)$$

   or just

   $$Pr(A \cap B) = Pr(A) \cdot Pr(B|A)$$

   The probability $Pr(B|A)$ is the conditional probability of B given A, meaning the probability of B knowing that A has occurred.

From the laws above the famous *Bayes' theorem* can be derived:

$$Pr(A|B) = \frac{Pr(B|A) \cdot Pr(A)}{Pr(B)}$$

The theorem is important since it makes it possible to compute the conditional probability relating two events from the "reverse" conditional probability. Note that the probability of event A given event B is generally different from the probability of B given A, but that there is a definite relationship which Bayes' theorem states. The various probabilities is further explained below:

- $Pr(A)$ is the marginal probability of A, also called A's prior probability.
- $Pr(A|B)$ is the conditional probability of A given B, as explained earlier, and is often referred to as the posterior probability.
- $Pr(B|A)$ is the likelihood of A given fixed B, $L(A|B)$
- $Pr(B)$ is the marginal probability of B, and in this case it acts as a normalizing constant.

To sum up, Bayes' theorem may be written as

$$posterior = \frac{likelihood \times prior}{normalizing\ constant} \propto likelihood \times prior.$$

One important application of probability theory is Bayesian networks [49], a network structure representation that encodes the joint probability distribution for its domain. Bayesian networks consists of a set of variables and a set of directed edges between them, where each variable has a finite set of mutually exclusive states, and together with the directed edges they form a directed acyclic graph (DAG). Combined with the structural information of the networks, conditional probabilities make it possible to reason more efficiently. Bayesian networks are further investigated in Chapter 3 (learning) and Chapter 4 (reasoning) respectively.

2.2 Expert systems

The traditional view of expert systems can be summarized as put by [20]:

> "Expert systems are attempts to crystallize and codify the knowledge and skills of one or more experts into a tool that can be used by non-specialists."

In the case of ConocoPhillips' production processes, it is not necessarily the case that experts are in possession of the desirable knowledge. Hence, it is also a goal to discover new knowledge of the production process. Anyway, without stressing the use of the term *expert*, such systems consist of two parts; a knowledge base and an inference engine.

The knowledge base contains domain-specific knowledge of a problem. *The inference engine* is the system's brain, in that it reason about the information in the knowledge base together with information available for the application at hand, in order to reach some conclusion. Having this in mind, the quality of the system is limited to the quality of the knowledge base. If the contents of the knowledge base is poor, this will reflect the quality of the inferences. A clean separation of both components is preferable in order to be able to improve the knowledge base if more information is acquired, and it also facilitates learning from the experience of making mistakes [20].

According to Onisko et al. [45] there exists two major classes of expert systems, known as *rule-based systems* and *probabilistic expert systems*. Rule-based systems are based on logical rules. The symbolic reasoning mechanism has two main methods: (1) *Forward chaining*, that is, making valid deductions from a given set of assertions, and (2) *Backward chaining* that is the reverse operation (determine whether assertions exist that can validate a conjectured property).

2.2.1 Rule-based systems

One of the most widely discussed rule-based expert systems is MYCIN [53], a clinical consultation program. It is based upon the concept of Certainty factors (CF) whose simple and intuitive appeal makes it interesting to briefly summarize, while at the same time bringing a historical perspective to the forthcoming discussions.

With CF, the consequent of rules with the form

$$IF\ \langle evidence \rangle\ THEN\ \langle hypothesis \rangle$$

is assigned a numerical weight in the interval $[-1, 1]$. The CF is computed from the degree of belief, MB, and the degree of disbelief, MD, that are defined as follows:

$$MB = \begin{cases} 1 & \text{if Pr(h)=1} \\ \frac{max[Pr(h|e), Pr(h)] - Pr(h)}{1 - Pr(h)} & \text{else} \end{cases}$$

$$MD = \begin{cases} 1 & \text{if Pr(h)=0} \\ \frac{min[Pr(h|e), Pr(h)] - Pr(h)}{Pr(h)} & \text{else} \end{cases}$$

The formula for CF is then $CF = \frac{MB - MD}{1 - min(MB, MD)}$, which gives a number between -1.0 (total disbelief) and 1.0 (total belief). It is tempting to interpret CF as statements of conditional probabilities, but as pointed out by Spiegelhalter et al. [20] this can lead to inconsistency in that an arbitrary set of rules might not be compatible with any overall probability distribution. If, on the other hand, this is the case, it is not certain that the distribution is unique.

2.2.2 Probabilistic expert systems

Probabilistic expert systems, also called normative systems, are cross-over products from the field of AI and statistics. They are graphical tools for knowledge representation and are based on the mathematical foundation of probability theory. Even though the knowledge base probabilities are given by an expert, reasoning is done applying a mathematical formalism. The idea of using the graphical structure to make inference modular, hence introducing the important property of local computations, was probably first introduced by Pearl [48]. One of the most prominent tools for this kind of expert systems are Bayesian networks. As mentioned in Chapter 2.1, a more thorough investigation of Bayesian networks is given in the succeeding chapters. Nevertheless, a simple example concludes this chapter in order to illustrate some important properties of Bayesian networks, as they play an important role in the understanding of probabilistic expert systems.

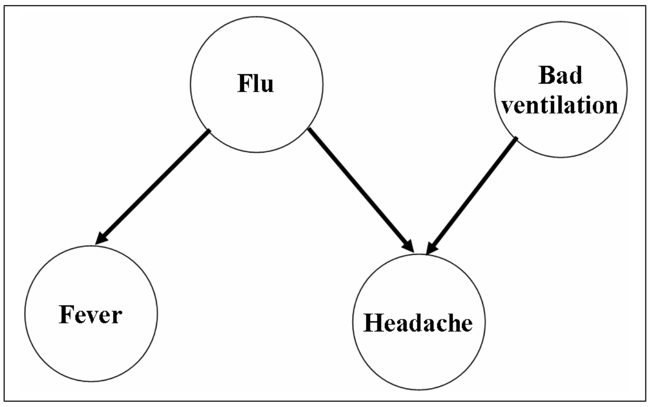


Figure 2.1: Bayesian network for the headache domain

Suppose one wish to reason about headache, and that there are two possible causes, namely the flu or a malfunctioned ventilation system at work. Having the flu can also cause fever. A Bayesian network[3] for this simple domain is shown in Figure 2.1. Table 2.1 shows the conditional probabilities for having a headache, while Table 2.2 shows the conditional probabilities for having fever. In addition, the probabilities for having the flu or that the ventilation system is malfunctioned are 0.01 and 0.15 respectively.

Flu	**Bad ventilation**	**Headache=yes**	**Headache=no**
Yes	Yes	0.99	0.01
Yes	No	0.90	0.10
No	Yes	0.30	0.70
No	No	0.05	0.95

Table 2.1: Conditional probability table for having a headache

Flu	**Fever=yes**	**Fever=no**
Yes	0.80	0.20
No	0.05	0.95

Table 2.2: Conditional probability table for having fever

To calculate the marginal probability for having a headache, i.e. the probability of headache regardless of other events, one has to sum over all combinations of states for fever, flu and bad ventilation. Thus,

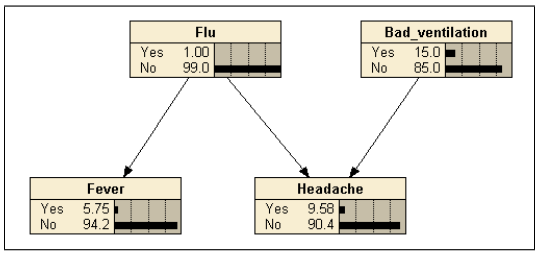
$$Pr(Headache = yes) = \sum_{Fl}\sum_{BV}\sum_{Fe} Pr(Headache = yes, Fever, Flu, Bad\ ventilation) = 0.0958.$$

[3]Figures in this examples are screen shots from Netica [43], a software package for Bayesian networks, decision nets and influence diagrams. Calculations are also made using Netica.

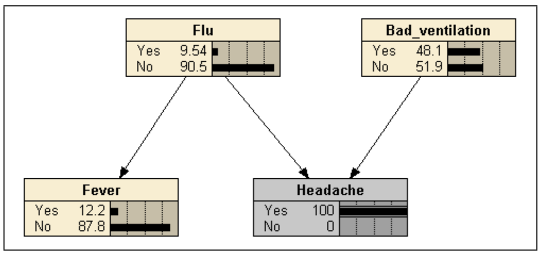
The interested reader can find the full calculation in Appendix A.1.

If one day headache occurs, this observation is stated in the Bayesian network in order to update the other events. To find the most likely cause, one most carry out the marginalisation process to obtain $Pr(Flu = yes|Headache = yes) = 0.0954$ and $Pr(Bad\ ventilation = yes|Headache = yes) = 0.481$.

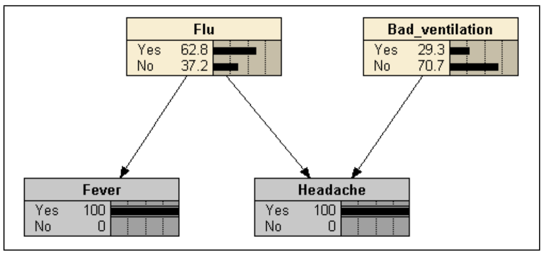
Suppose that later the same day, fever is observed. This changes the probabilities to 0.628 and 0.293 respectively. Thus, observing fever has *increased* the belief in flu causing the headache and at the same time *decreased* the belief in a malfunctioned ventilation system as the cause. This effect is called *explaining away* since a tentative conclusion (bad ventilation is the most probable cause for headache) is withdrawn on the basis of further information (observing fever causes the flu to be the most probable cause for headache). Illustrations of the probability evolution just explained are given in Figure 2.2.



(a) Initial probabilities



(b) Probabilities after observing headache



(c) Probabilities after observing fever

Figure 2.2: Illustrations of probability evolution for the headache example

2.3 Association rule mining

The task of association rule mining was introduced in 1993 [1] and has since been used in many application domains. Looking back to the starting point, the motivation for mining association rules was the great amount of *basket data*; items purchased on a per-transaction basis. Finding association and patterns between products can be an important knowledge when it comes to decision making and marketing.

Adopting the formal definition given by [28], let $\mathcal{I} = \{x_1, ..., x_n\}$ be a set of distinct literals, called items (e.g. articles). Also, let the database $\mathcal{D}$ be a multi-set of subsets of $\mathcal{I}$ where each $T \in \mathcal{D}$ is called a transaction (e.g. if T consists of articles bought by one particular customer, then $\mathcal{D}$ consists of all such purchases). A set of items, $X \subseteq \mathcal{I}$, is referred to as an *itemset*, or more often a k-itemset where k = $|X|$. An association rule is on the form $X \rightarrow Y$ where X and Y are itemsets and $X \cap Y = \emptyset$ (following the example so far, X can be bread and Y milk). Two important measures are defined from this:

1. **Support:** A transaction $T \in \mathcal{D}$ supports an itemset $X \subseteq \mathcal{I}$ if $X \subseteq T$ holds. The support of the rule $X \rightarrow Y$ is the percentage of transactions in $\mathcal{D}$ that contain X and Y. This is defined as

$$supp(X \rightarrow Y) = \frac{\#(X \text{ and } Y)}{\#(T \in \mathcal{D})} \tag{2.1}$$

2. **Confidence:** The confidence of a rule $X \rightarrow Y$ is the percentage of transaction in $\mathcal{D}$ containing X that also contain Y. This is defined as

$$conf(X \rightarrow Y) = \frac{supp(X \rightarrow Y)}{supp(X)} \tag{2.2}$$

An itemset is called *frequent*[4] if it satisfies a specified minimum support threshold. Rules that support both a minimum support threshold and a minimum confidence threshold are called *strong* [50]. As mentioned by several authors (e.g. [42] and [28]), two main problems exists for association rule mining: (1) low algorithm performance and (2) results including a vast number of non-interesting rules. *First*, the theoretical number of rules grow exponentially with $|\mathcal{I}|$. The support and confidence measures are often used to split the task into two sub-problems [1]. A large number of algorithms have been developed to efficiently prune the search space and the Apriori algorithm [2] is maybe the most famous. The algorithm takes advantage of the fact that all non-empty subsets of a frequent itemset must also be frequent. By joining frequent (k-1)-itemsets to generate candidate k-itemsets, pruning the candidate set can be done by deleting those containing any subset that is not frequent. *Second*, the number of possible rules generated from the final set of frequent items can be quite large, and even if it is not, finding truly interesting information is a non-trivial task. This problem introduces the problem of deriving robust and usable interestingness measures for association rules.

In addition to support and confidence, many alternative interestingness measures have been developed. Liu et al. [37] suggests two subjective measures; *unexpectedness* (an interesting rule is one that is unknown to the user or contradicts the user's existing knowledge) and *actionability* (an interesting rule is one that a user can do something with to his advantage). Klemettinen [34] et al. use *templates* which are pattern expressions that describe the form of rules that are to be selected or rejected.

Another measure is *lift* (first introduced as *interest* by Brin, et al. [11]) which is defined as

$$lift(X \rightarrow Y) = \frac{conf(X \rightarrow Y)}{supp(Y)} \tag{2.3}$$

A lift value greater than 1 indicate an increase in probability of the consequent given the antecedent (itemsets containing X tend to contain Y more often than itemsets that do not contain X), while a lift value smaller than 1 indicates a negative correlation between the items. If the lift value is (near) 1 then the items appear together as often as expected due to random chance.

[4]The term *large itemset* was originally used by [1]

2.4 Hypergraph theory

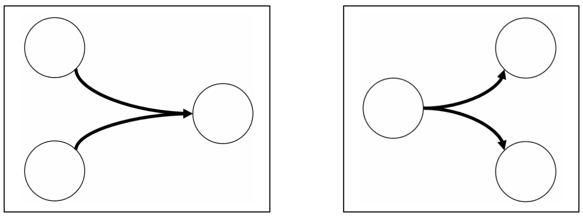
A review article on hypergraph theory, explaining terms, concepts and applications, is written by Gallo et al. [25], and acts as a guideline for what follows. Definition 1 defines the concept of *hypergraphs*:

Definition 1. *A hypergraph $H = (V, E)$ is a pair where V is the set of nodes*[5] *and $E \subset 2^V$. Each element of E is called a hyperedge.*

A *directed hypergraph* is a hypergraph with directed hyperedges or *hyperarcs*.

Definition 2. *A directed hyperedge (hyperarc) is an ordered pair, $e = (X, Y)$, of (possibly empty) disjoint subsets of nodes.*

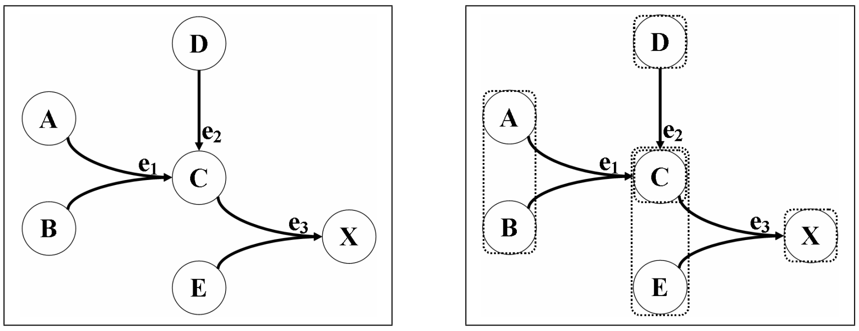
The node sets X and Y from Definition 2 are often referred to as tail ($T(e)$) and head ($H(e)$), respectively. A hyperarc is called a *B-arc* (Backward hyperarc) if $|H(e)| = 1$ (Figure 2.3a), and *F-arc* (Forward hyperarc) if $|T(e)| = 1$ (Figure 2.3b).



(a) B-arc (b) F-arc

Figure 2.3: Example of B-arc and F-arc

A *B-graph*, or B-directed hypergraph, is a hypergraph where all hyperarcs are B-arcs. Similarly an *F-graph* is a hypergraph whose hyperarcs are F-arcs, and a *BF-graph* is one with both F-arcs and B-arcs. As it will be clear from Chapter 3, it is assumed that rules will be on the form $X_1, \ldots, X_n \rightarrow Y$, i.e. the consequents are singletons. Furthermore, it will be argued that B-graphs are useful for representing such rules, hence only B-graphs will be considered further. An example of a B-graph is seen in Figure 2.4a.



(a) A B-graph example (b) B-graphs with hypernodes indicated

Figure 2.4: B-graphs

[5]The term "vertex" is often used instead of "node".

Since an edge can span several nodes, the notion of *hypernodes* must be defined, formally done in Definition 3.

Definition 3. *Given a B-graph B with hyperedges $\{e_1, \ldots, e_m\}$, the hypernodes induced by the hyperedge e_i are the tail $T(e_i)$ and the head $H(e_i)$ considered as single entities.*

As shown in Figure 2.4b the hypernodes corresponding to the hypergraph of Figure 2.4a are:
$\{\{A, B\}, \{C\}, \{D\}, \{C, E\}, \{X\}\}$. Note that e.g. E is a node, but not a hypernode.

Chapter 3

Learning network structures

Learning network structures from data is a non-trivial process where care must be taken as the process heavily influences how reasoning can be done. Other issues like the kind of input data, validity of correlations between elements and user requirements must also be considered. This chapter first derives assumptions made on input data. Rule syntax, rule semantics and rule cleaning are considered in the light of the overall process of utilizing data to predict oil production anomalies. Thereafter, the concept of Association Rules Network (ARN) is then described. ARN is a hyper-graphical model for representing rules whose consequents are singletons. Motivated by the lack of an overall framework with a purpose beyond simple use of the information encoded by the network structure, Bayesian networks are introduced next. Bayesian networks are among the most well-known tools for modelling variables and their probabilistic dependencies.

3.1 Assumptions on learning input

Being a part of an overall process, the network learning task receives its inputs from the preceding step of rule mining. Christiansen [17] studies this step in details and defines a rule syntax and rule semantics that can be used to target rules suited for prediction. It is argued that a restricted form of rule syntax is needed to ensure that rules unwanted for prediction are avoided. When combining rules in a reasoning structure, the rule syntax also becomes important in order to build a sound reasoning structure. Further, it is argued that the handling of time in a reasoning structure relies on a clear interpretation of the rules, and how they relate to the time of occurrences of events. Besides rule syntax and semantics, the network learning task depends on another part of the rule mining process called rule cleaning. Rule cleaning is the process of identifying rules which are suited for prediction from a large rule set. Normally, rule mining algorithms return a vast number of rules, so a process is needed for selecting the rules which are truly interesting and well suited for prediction.

In the following, assumptions made on the rule mining task are given. Using the notation of [17], *rule syntax and semantics* and *rule cleaning* are discussed separately.

3.1.1 Rule syntax and semantics

Rule syntax can be divided into three components: (1) Rule structure, (2) information about statistical strength of a rule and (3) information about time in a rule. **Rule structure** is the most important of the components when in comes to the isolated task of learning a network structure because some rule structures make the construction of a reasoning structure difficult. In general a rule, $\mathbf{X} \rightarrow \mathbf{Y}$, consists of two parts: the antecedent (left side, $\mathbf{X}$) and the consequent (right side, $\mathbf{Y}$). $\mathbf{X}$ and $\mathbf{Y}$ are sets of elements (events) with sizes $|\mathbf{X}| \geq 1$ and $|\mathbf{Y}| \geq 1$. This gives four combinations of the number of events in the antecedent and the consequent, as described in Table 3.1.

Rule structure	**Example**
single → single	A → B
single → multiple	A → B,C
multiple → single	A,B → C
multiple → multiple	A,B → C,D

Table 3.1: Possible rule structures

The previous work of Christiansen and Helland [18] states that the best suited rule structure for integration in a reasoning structure is rules on the form $multiple \rightarrow single$. The choice of discarding rules with multiple events on the consequent side is made mainly because they are difficult to interpret. In the case of using rules in a network for reasoning, having only one consequent element is more intuitive. Rules with a single event in the antecedent does not support combinations of events as a trigger, while rules that allow for multiple events in the antecedent make it possible to find combination of events that together contribute to a failure. Hence, the assumed rule structure is $multiple \rightarrow single$.

Information about statistical strength of a rule concerns what kinds of measures that are computed for each rule. Christiansen [17] recommends that *support*, *confidence* and *lift* are included in the rule syntax. He also uses these measures to derive other interestingness measures for each rule, hence the rule syntax should not be limited to the three mentioned only.

The last aspect of the rule syntax is the **amount of time information** included. It is here argued that the rule syntax should not limit the possible reasoning mechanisms by leaving out information about time. Theoretically it is possible to derive statistical distributions for the time between events in a rule. A question must at the same time be asked what kind of information the operator will find useful. Adding too much information of time will not only complex the reasoning scheme, it can also confuse the users of the system. It is therefore assumed that information about the minimum time and maximum time between the antecedent and the consequent is included in the rule syntax.

When it comes to rule semantics, it has already been stated that the handling of time in a reasoning structure relies on a clear interpretation of the rules, and how they relate to the time of occurrences of events. Given a rule structure on the form $multiple \rightarrow single$ the required temporal order of events imposed by a rule becomes important in two ways; the order imposed between the antecedent and the consequent (rule ordering), and the order imposed among the events on the antecedent side of the rule (antecedent ordering). This gives four possible combinations of ordering among events, as seen in Table 3.2.

Consider first rule ordering: A parallel rule imposes no temporal order between the antecedent and the consequent, while a serial rule imposes a total order between the antecedent set and the consequent,

Rule ordering	**Antecedent ordering**
parallel	parallel
parallel	serial
serial	parallel
serial	serial

Table 3.2: Possible ordering of events

requiring that events in the antecedent occur before the consequent. As an example, consider the rules $A \rightarrow B$ and $B \rightarrow A$. With a serial rule these two rules differ, in opposite to a parallel ordering where they constitute the same rule. Considering the case of predicting failures in oil production, which is an ongoing time-spanning process, it is only interesting to predict events forward in time. Hence, rules given as input to the learning process are assumed to be serial.

According to Table 3.2 also the antecedent elements can have a parallel or a serial ordering. Take as example the two rules $A, B \rightarrow C$ and $B, A \rightarrow C$. Only when the antecedent is defined to be serial these two rules differ. The consequence of using a parallel antecedent is that a belief in the specific ordering of events causing the consequent is repressed. This could lead to loss of information, but according to Maintenance Optimisation Engineer Fredrik Høymer Fossan at ConocoPhilips, the domain relations mainly correspond to the occurrence of certain events within a specific time interval [24]. Since a parallel antecedent makes less restrictions on the type of network that can be used, it is assumed that the input rules conform to this property.

Having assumed that rules will have the structure $multiple \rightarrow single$ with a serial rule ordering and a parallel antecedent ordering, it is possible to define the notion of time. Figure 3.1 explains the temporal relations to be considered, using the rule $X_1, X_2 \rightarrow Y$ as an example.

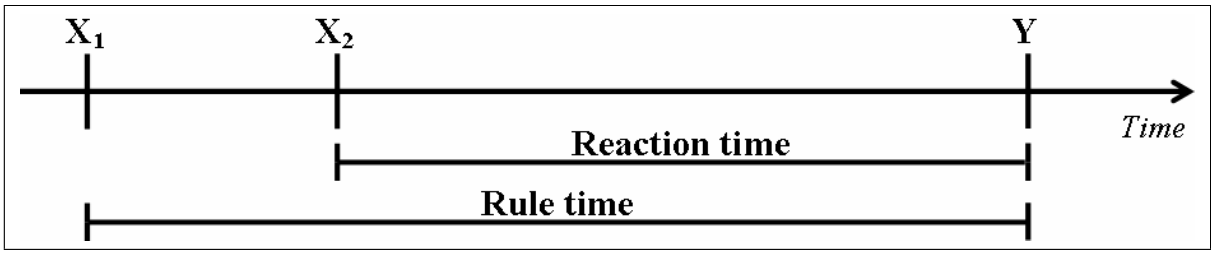


Figure 3.1: Definition of temporal relations using the rule $X_1, X_2 \rightarrow Y$

From this four time limits are defined:

- **Minimum reaction time limit:** Part of the rule syntax, but should conform to the minimum reaction time for the operator.
- **Maximum reaction time limit:** Part of the rule syntax.
- **Minimum rule time limit:** Same as minimum reaction time limit.
- **Maximum rule time limit:** Defined by the width of the sliding window [17].

Table 3.3 summarizes the assumptions made on rule syntax and semantics.

Rule structure	multiple $\rightarrow$ single
Statistics	$Support$, $confidence$ and $lift$ as a minimum, but other measures can also be included.
Time information	$Minimum$ and $maximum$ time between the last event in the antecedent and the consequent event.
Rule ordering	Serial
Antecedent ordering	Parallel
Reaction time limit	Lower and upper bound are a part of the rule syntax, but the lower should conform to the minimum reaction time for the operator.
Rule time limit	Lower bound is the same as minimum reaction time, and upper bound is defined by the width of the sliding window used for rule mining
Example	$A, B \rightarrow C$ *(min: 10, max: 60)* *(sup: 7%, conf: 90%, lift: 7)*

Table 3.3: Assumptions made on rule syntax and semantics

3.1.2 Rule cleaning

Rule mining in general produces a vast number of rules, and a difficult problem is how to find the small part of the resulting rule set that is really interesting [52]. The process of finding the set of interesting rules is here known as *rule cleaning*. As the network learning process itself has no influence on the rule cleaning process, some assumptions must be made on the final set of rules:

- The rules consists of events that are understandable to the user.
- The rules are relevant and interesting to such an extent that no further pre-processing must be done on the rule set.
- An event that represents an anomalous situation that is to be avoided, is assumed to occur as consequence in one or more rules.

A particular situation that must be commented is when two rules have the same consequent. Take as an example the rules $A \rightarrow C$ and $B \rightarrow C$; what about the rule $A, B \rightarrow C$? According to the rule cleaning assumptions this depends on whether the rule is seen as interesting or not. This heavily restricts the kind of reasoning mechanisms that can be suggested, so it is assumed that communication with the rule mining step is possible so that extra information can be retrieved if needed.

3.2 Association rules network

Given the goal of prediction, the intuitive way of seeing the problem of learning a network structure from a set of rules is to build it recursively from the goal node. ARN, as introduced by Chawla et al. [14], is a hyper-graphical model for representing rules whose consequents are singletons. More specific, rule sets are represented and visualized as flow networks, using a hyper-graphical notation with weighted edges.

Definition 4 is that of [15] with two exceptions: (1) The goal node (item) z does not has to be frequent, and (2) the hyperedge weights do not have to be the confidence of the rules. These two exceptions follows directly from the overall solution where the preceding step is that of discovering dependencies between events. The rules of interest do not have to be frequent (it might be rare) and confidence is just one of several possible measures of interestingness (so it is not excluded either).

Definition 4. *Given a set of association rules R and a goal item z which appears as a singleton in a consequent of a rule $r \in R$. An association rules network, $ARN(R, z)$, is a weighted B-graph such that*

1. *There is a hyperedge which corresponds to a rule r_0 whose consequent is the singleton item z.*
2. *Each hyperedge in $ARN(R, z)$ corresponds to a rule in R whose consequent is a singleton. The weight on the hyperedge is some interesting measure of the rule.*
3. *Any node $p \neq z$ in the ARN is not reachable from z.*

An important property is the definition of a *goal node*, which every other node in the ARN are B-connected to. It corresponds to the notion of a failure node for the oil production process. If there exists several state of failures modelled as an event, an ARN must be built for each of them.

The earliest ARN learning algorithm [14] does not remove cycles. The authors note that a solution could have been to add a constraint that a node which has served as a consequent cannot be an antecedent. However, they reject that solution as that would destroy the uniqueness of the network. In order to exclude cycles from an ARN while retaining reachability to the goal node, Chawla et al. [15] defines the notion of node level, as introduced in Definition 5. Note that the definition of non-goal node level is here slightly changed on the original article: A "+ 1" is added to in order to increase the count (if not, all nodes would have been given the level zero).

Definition 5. 1. *The level of the goal node is zero.*

2. *The level of a non-goal node v, is defined as*
   $l(v)$ = *min* $\{l(u)|\exists e \text{ such that } v \in T(e) \text{ and } u = H(e)\}$ + *1*
3. *The level of a hypernode c is defined as $l(c)$ = min$\{l(s)|s \in c\}$*

As an example, consider Figure 2.4b in Chapter 2.4, in which $l(X)$ = 0, $l(E)$ = 1, $l(C)$ = 1, $l(D)$ = 2, $l(B)$ = 2, $l(A)$ = 2, $l(\{C, E\})$ = 1 and $l(\{A, B\})$ = 2.

From the definition of node level, the definition *reverse hyperedges* [15] can be derived, which is important since they generate redundant paths from a node to the goal.

Definition 6. *A hyperedge e in an ARN is called a reverse hyperedge if $l(T(e)) < l(H(e))$.*

3.2.1 Learning ARN

Informally, an ARN is learnt by first fixing an element c and finding all rules whose consequent is c. In the next stage the antecedents of the selected rules play the role of consequents. [15] introduces a breadth-first like algorithm for learning an ARN from a rule set R and a goal node c. It is an improved version of the one in [14] in that it removes circular paths and redundant and backward edges. The prevention of redundancies (cycles and reverse hyperedges) is based on two conditions:

- **Condition 1:** A node which has served as a consequence during ARN generation cannot be an antecedent across levels for a rule whose consequent is at a higher level.
- **Condition 2:** The goal attribute can appear with only one value in the ARN, namely the one in the goal node.

Consider the following set of rules:

$A \rightarrow B$	$A \rightarrow C$
$B \rightarrow D$	$C \rightarrow E$
$C \rightarrow F$	$D \rightarrow A$
$D \rightarrow E$	$E, F \rightarrow X$

Figure 3.2 shows the ARN learnt from the rules. The dashed edge from D to A, corresponding to the rule $D \rightarrow A$, is a reverse hyperedge and will not be a part of the final network. The learning algorithm is further discussed in Chapter 3.2.2.

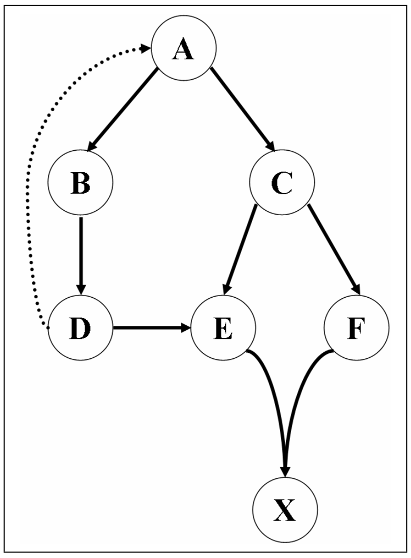


Figure 3.2: ARN with reverse hyperedge

The concept of ARNs has not been investigated by many authors, and a review of the publications within the field has not revealed any suggestions of how to add time semantics to the networks. Time semantics should not complicate the network in such a way that standard propagation techniques[1] cannot be used. Since both nodes and edge weights must be updated continuously as events occur, an idea is to use some sort of overall time mechanism in order to adjust the weights according to the rules time of validity. This would not alter the propagation techniques, as required. Chapter 4.1.4 continues the discussion on temporal information in ARNs.

[1]The statement "standard propagation techniques" refers to the propagation techniques that belongs to each individual type of network when no extra semantics are added to nodes or edges.

3.2.2 Improving the ARN algorithm

The ARN learning algorithm described in the literature [15] has some flaws that must be corrected in order to reach the goal of learning a network structure from a set of rules. In Algorithm 1 these flaws are taken care of, and each correction is explained in the following. The algorithm assumes that the goal node occurs as a singleton consequent item in at least one rule, and has a time complexity of $O(kn)$ where n is the number of distinct items and k is the number of rules.

- Line 1: Initialization of $visited[i] = \infty, \forall_i$ and $level[i] = \infty, \forall_i$ is added.
- Line 2: For the consequent, c, the level is initialized to 0 ($level[c] = 0$).
- Line 7: The variable s is never updated in the original algorithm. It is assumed that the intended meaning is to avoid edges between nodes constituting a hypernode (with two or more nodes). For the temporal nature of the problem domain this is not desirable, so the variable is removed. A consequence is that the function $Rules.getRules(R, u, s)$ is transformed to $Rules.getRules(R, u)$. In addition an assumption is done on the rule base that it does not contain rules where the consequent element is also a part of the antecedent.
- Line 9: **a** is defined precisely to be a set of elements (possible of size 1). In line 10 it is the *set*, which represents the notion of a hypernode, that is set to have $level = \infty$. For an explanation of the difference between a node and a hypernode see Definition 3 of Chapter 2.4.
- Line 9: the $Rules.getAntecedents()$ is called with argument r, not u as in the original algorithm.
- Line 30: Cycles between hypernodes on the same level are not necessarily removed based on confidence, but based on a user selected interestingness measure. Note that such a measure cannot be symmetric in order to distinguish rules like $A \rightarrow B$ from $B \rightarrow A$.
- Line 35: The variable u is updated at the end of each loop. u is set to be the next element of the queue q of elements that have played the role as antecedent. This queue is thus *first-in, first out* (FIFO).

ARN is an intuitive and simple concept, but it does not conform to a greater framework with a purpose beyond simple use of the information encoded by the network structure. A more consistent concept should be investigated where the network structure and network reasoning are close connected. This is the subject for Chapter 3.3.

```
**Data**: Rules $R$, Consequent $c$
**Result**: A directed hypergraph $G$ representing an ARN which flows into $c$
/* $visited[i] = 1$ if $i$ has been visited as a consequent */
Initialize $visited[i] \leftarrow 0$, $level[i] = \infty, \forall_i$;
$level[c] = 0$;
$visited[c] \leftarrow 1$;
$u \leftarrow c$;
Add $u$ to queue $q$;
repeat
    $RuleSubset \leftarrow Rules.getRules(R, u)$;
    /* Get all rules whose consequent is $u$ */
    for all rules $r \in RuleSubset$ do
        a $\leftarrow Rules.getAntecedents(r)$;
        $level[\mathbf{a}] = \infty$;
        for all elements $w \in$ a do
            if level[w] < level[a] then
                $level[\mathbf{a}] = level[w]$;
            end
        end
        if $level[a] \geq level[u]$ then
            for all elements $w \in$ a do
                if visited[w] = 0 then
                    Add $w$ to $q$;
                    $visited[w] = 1$;
                    $level[w] = level[u] + 1$;
                end
            end
            $G.addHyperEdge(r, u)$;
            /* Directed hyperedge flowing into $u$ */
        end
    end
    if q is empty and G is singleton then
        return $G = \emptyset$;
    else if q is empty and G is not singleton then
        $G.removeLevelCycles()$;
        /* Remove level cycles based on interestingness measure */
        return $G$;
    end
    end
    Delete $u$ from $q$;
    $u \leftarrow next(q)$;
until false;
```

Algorithm 1: The corrected ARN learning algorithm

3.3 Bayesian networks

A Bayesian network is a network structure representation that encodes the joint probability distribution for its domain and consists of a set of variables and a set of directed edges between them. Each variable has a finite set of mutually exclusive states, and together with the directed edges they form a directed acyclic graph (DAG). For each variable Y with parents $X_1, ..., X_n$, the conditional probability table (CPT) $P(Y|X_1, ..., X_n)$ is given. As noted by [31], the definition of a Bayesian network does not refer to causality, so it is not required for the links to represent causal impact. When learning Bayesian networks it is often easier for humans to think of the connection between two nodes as causal. The emphasis should, however, be put on avoiding to include conditional independences which do not hold in the real world. Figure 3.3 shows an example of a Bayesian network. It encodes the joint probability for the variables represented by the nodes A-H, in addition to a set of independence assumptions as explained next.

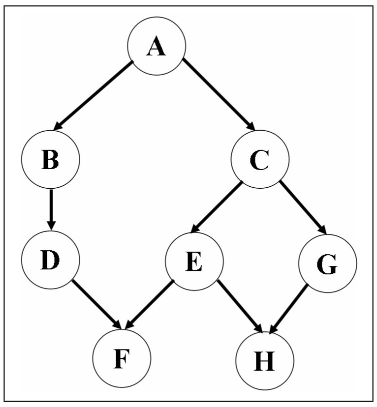


Figure 3.3: Bayesian network example

A complete specification of a probability distribution requires vast numbers. As an example, consider n binary random variables; the complete distribution is specified by $2^n - 1$ joint probabilities. The built-in independence assumptions for Bayesian networks can reduce this number dramatically. This is due to the fact that in a Bayesian network, each variable is independent of its non-descendants in the graph given the state of its parents. In Figure 3.3 this leads to the conclusion that node F is independent of every other node given D and E, while node E is independent of A-D and G given C. Thus, the joint probability distribution can be uniquely determined by the local conditional distributions, i.e. the conditional distribution of a variable given its parents.

3.3.1 Learning Bayesian networks

Research on Bayesian network learning gained speed by the publishing of "Learning Bayesian networks: The combination of knowledge and statistical data", an article by Heckerman et al. [27] that soon was considered required reading for Bayesian network researchers. It describes a Bayesian approach for learning Bayesian networks from a combination of user knowledge and statistical data. Figure 3.4 shows a view of the Bayesian network learning task.

The task of learning a network from a set of restricted association rules, differs from the method of [27] in two ways:

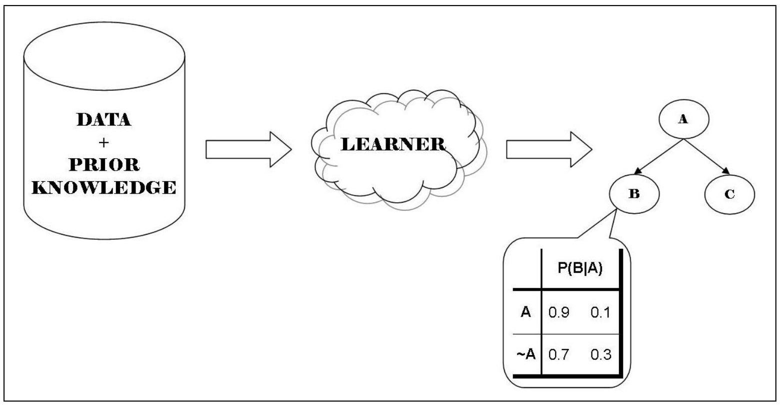


Figure 3.4: Bayesian network learning

- No prior knowledge of the network structure exists.
- The data to be learnt from is not stored in database form, but consists of rules.

It is now suggested that the network structure learnt by Algorithm 1 can be used as a basis for learning a Bayesian network. To do this some adjustments must be made as discussed below:

- **Network structure:** In general, each hyperedge e has to be replaced by single edges. This is done by removing e and then add one edge between each node $v \in T(e)$ and $H(e)$. By doing this the network structure becomes a DAG as requested. Note that a node can be a part of several hyperedges leading to the same consequence, so a test must be made in order to avoid redundancies. As an example consider the rules $A, B \rightarrow X$ and $B, C \rightarrow X$. The process of replacing hyperedges by single edges will create two edges from B to X, but only one is necessary.
- **Conditional probability tables:** The CPT for each variable Y with parents $X_1, \ldots, X_n$ must be specified. From the process of mining restricted association rules is it possible to derive the required numbers by calculating the confidence for the necessary combination. As an example consider the rule $A, B \rightarrow C$. The CPT for C includes: $conf(A, B \rightarrow C)$, $conf(A, \neg B \rightarrow C)$, $conf(\neg A, B \rightarrow C)$ and $conf(\neg A, \neg B \rightarrow C)$.
- **Composite[2] rules:** As mentioned in Chapter 3.1.2 a situation may arise that two rules have the same consequent, e.g. $A \rightarrow C$ and $B \rightarrow C$. However, Bayesian networks do not distinguish $\{A \rightarrow C, B \rightarrow C\}$ from $\{A, B \rightarrow C\}$, and the CPT for C will be as above for both situations. If the correspondent composite rule ($A, B \rightarrow C$) is not a part of the input rule set, communication with the rule mining process is needed to obtain the full CPT. In the opposite case redundancies will be created, so a simple test must be performed while transforming hyperedges: For each hyperedge e, remove any single edge from $v \in T(e)$ to $H(e)$ before accomplishing the hyperedge replacement.

Considering the challenge of composite rules as discussed above, it might be tempting to delete a general rule (e.g. $A \rightarrow C$) from the rule set when a more specific one exists (e.g. $A, B \rightarrow C$). However, this cannot be done due to the cross-level restriction of the network learning algorithm. Recall that the level

[2]A rule is called *composite* if the number of antecedent elements is of size two or greater.

of a hypernode is defined to be the lowest level of its single elements. A composite rule might be skipped from the network due to the cross-level restriction, while at the same time it is possible that a single rule whose antecedent is of a higher level will be a part of the network. This justifies why general rules must exist in the presence of more specific ones.

3.3.2 Time modelling in Bayesian networks

Bayesian networks do not provide a direct mechanism for representing temporal dependencies. In [18] an overview of Bayesian network variants that incorporate the notion of time is given. None of Temporal Bayesian networks (TBN) [57], Dynamic Bayesian Networks (DBN) [39, 33] or hidden Markov models (HMM) [51] are deemed suitable for the overall solution proposed. A suggestion is given to alter the node semantics such that a node's conditional probability table reflects the probability that the event will occur within some time if one or more of its parents occur. This approach has in fact been introduced by Arroyo-Figueroa and Sucar [8, 5, 4] as Temporal Nodes Bayesian Network (TNBN), and later as Temporal Bayesian Network of Events (TBNE) [7]. They apply the method to the dynamic domain of fossil power plant diagnosis, among others.

As pointed out by the founders of TBNE it is complicated to extend Bayesian networks semantics to deal with temporal relationships. The main problem is to represent each node with its dependence relationships over multiple points of time. The motivation behind the development of TBNE is the fact that in many cases there are few state changes in the temporal range of interest. At the same time, it is also expedient to make a simple network where standard probability propagation techniques for diagnosis and prediction can be used. The goal is to represent a complex system evolving over time, with the following main properties: (1) Given evidence about the past and present state of the system, predict the system's future state, and (2) given a future state determine the most probable cause.

A TBNE is a Bayesian network of events in discrete time where each temporal node represents an event or state change of a variable, and the arcs represent causal-temporal relationships between the nodes. A *temporal node* is defined as a set of ordered pairs (σ, τ) where σ is the state or value variable and τ is the time interval associated with each state variable attribute. The conditional probability distribution for each node is defined as the probability of each ordered pair (σ_i, τ_i) given the ordered pair of its parents (σ_j, τ_j). There is a default state of no change that corresponds to the initial value (or the normal value), associated to the full temporal range of the node. It is pointed out that while previous approaches are based on point models of time, and as such require that events occur instantaneously, it is often more natural to consider events taking place over time intervals.

In a TBNE temporal information is relative, meaning that there is not an absolute temporal reference. The temporal intervals on each node are relative to its parents. Relative times are transformed to absolute based on the timing of the observations; an initial event is used as a temporal reference for the other events. In other words, the actual timing of the events represented in the network is dynamic.

Figure 3.5 shows a TBNE example as presented by [4] which models a drum level system in a fossil power plant (detailed knowledge about this process is not necessary in order to understand the following). Consider the node labelled DHL (Drum Level High condition) and its parent, FWF (Feedwater Flow increment). Table 3.4 gives the probability of the state and corresponding time interval for DHL given the state and time interval of FWF. Remember that the time intervals are relative.

In order to learn a TBNE, the arguments from Chapter 3.3.1 still hold. In addition, time intervals must

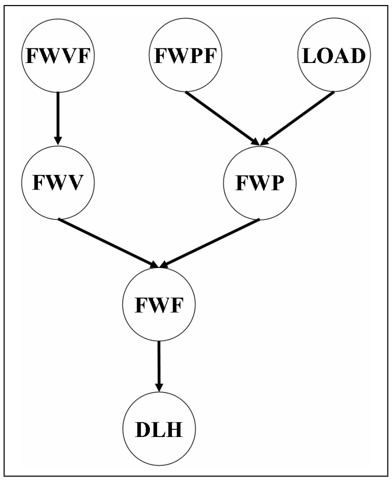


Figure 3.5: A TBNE example

FWF	**True, [10–27]**	**True, [27–135]**	**False, [10–135]**
True, [25–144]	92%	5%	3%
True, [144–248]	8%	91%	1%
False, [10–27]	2%	2%	96%

Table 3.4: CPT for the DLH node

be defined. By altering the node semantics as just described, the number of states of a node needed to represent the state change of the process potentially increases. In other words, a consequence is an additional number of assignments in the joint probability distribution, increasing the complexity. However, based on the preceding rule syntax discussion it is clear that for the overall solution suggested only two intervals will be defined for each node: *True, [Minimum time, Maximum time]* and *False, [0, Maximum rule time limit]*. This is information available to the learning process.

As previously discussed, a situation may occur that a composite rule is "missing" in the input rule set. In the same way as for CPT-values, time intervals must be retrieved from the mining process. This shows the need for an overall strategy when developing a complete solution, i.e. the three recommended steps cannot be done in total isolation. Note that the Bayesian learning approach briefly mentioned in Chapter 3.3.1 does not include support for temporal information. For the process of learning a TBNE, [7] uses an expert to define both the network structure and the time intervals. Using this method restricts the size of the system to be modelled, and enforces the need for a domain expert with thorough knowledge of the processes. To turn the situation, restricted association rules might be seen as a replacement for expert knowledge. In addition, the rule mining process might find associations unknown to the user.

3.3.3 Issues by combining Bayesian networks and association rules

Other authors have applied Bayesian networks and association rules together for various purposes. Fauré et al. [23] uses a Bayesian network together with association rules in an iterative process. A network is initially built by considering a priori expert domain knowledge. When mining rules, this network is used to exclude rules describing known dependencies. The association rules remaining are then used to update the Bayesian network, using an expert-driven annotation process. Motivated by the fact that in data mining the patterns which diverge the most from the background knowledge are deemed most interesting, Jaroszewicz and Simovici [30] also use a Bayesian network to represent a priori knowledge. They, however, use this knowledge to measure the degree of interest for mined itemsets. The most interesting itemsets are iteratively used to update the background knowledge. A third application of the combination of association rules and Bayesian networks are given by Lamma et al. [35]. They propose an improvement for the Simple Learning Algorithm (SLA) [16], which is a well known algorithm for learning Bayesian networks, by using association rules to infer the initial structure of the network.

A review of the publications within the field made by [18] did not reveal any common applications of Bayesian networks as decision structures based solely on mined association rules. The reason for this is twofold: *First*, there is a disagreement about the concept of causality [32]. Bowes et al. [10] argue that association rules are semantically weak, meaning that such rules do not necessarily imply any deeper relationships between the involved variables. Further, they conclude that mined rules must be evaluated by a domain analyst, something which is both time-consuming and difficult. As an alternative, causal inference algorithms (like the Bayesian approach to Bayesian network learning) are proposed. It is claimed that where association rule generation techniques find surface associations, causal inference algorithms identify the structure underlying such associations. Tse and Liu [56] agree that causal inference algorithms always extract relationships that are stronger than association rules, since their elicitation is based on rigid statistical testing. A problem is, however, that the tests can be too rigid that some subtle, but novel rules would be rejected during the testing process. Association rules, on the other hand, can be mined with novelty in mind (e.g. setting a low support threshold). Another problem, pointed out by Karimi and Hamilton [32], is that Bayesian networks find causality even in domains where the existence of causal relations itself is a matter a debate. An example given by [54] is the rule "*Minister* is caused by *Prime*", found by analysing words in political texts using Bayesian network methods.

It should be clear from the discussion above that the concept of causality is not a one-sided issue since there is a balance between the strength of relationship and novelty. Considering the overall process described in Chapter 1.3.3 there is a belief in the concept of rule interestingness, i.e. novelty should be weighted at least as much as strength of relationship. When it comes to network learning and reasoning the focus is also on the relationship between the rules as a set, not on each individual rule.

Second, association rules and Bayesian networks are in fact examples of methods from different research traditions within the field of data mining, as explained by Hollmén et al. [29]: *Probabilistic modelling* consists of what can be called global techniques and views data mining as the task of approximating the joint distribution. The idea is to develop modelling and description methods that incorporate an understanding of the generative process producing the data. The other tradition can be described as *the technology of fast counting* [38] and can be seen as a collection of local techniques. The goal is often to discover frequently occurring patterns. Each pattern and its frequency indicate only a local property of the data, and a pattern can be understood without having information about the rest of the data. Association rules belong to this tradition.

Mannila [38] explains that the process of mining association rules can be seen as a search for simple descriptions that are true for a reasonably large fraction of the data set. The author argues, however, that even though each pattern can be understood in isolation, the algorithms are complete. In other words they find all patterns from a class of patterns that satisfy certain conditions, hence also some global information is provided. Simply speaking, the association rules themselves are local, but the collection of rules gives global information of the data set.

As noted by [29, 38] there are in general many distributions that can explain the observed frequent sets. Suppose the frequencies $f(X)$ for all subsets X are known such that $f(X) \geq \sigma$. For $\sigma = 0$ the frequencies of frequent set determine the distribution uniquely; however, $\sigma = 0$ means $2^{|R|}$ frequent sets, and therefore the approach is infeasible. For $\sigma > 0$ methods like inclusion-exclusion [47] can be used, but that is beyond the scope of this discussion. The important realisation is that a collection of rules provides global information that might be utilized in some manner.

Chapter 4

Reasoning in network structures

Having built a network from a set of rules, the next step is to define how to reason in such a representation. The process of *reasoning* is tightly connected to *inference*. The interpretation of the latter term is domain dependent: In statistics, it is the process of drawing conclusions about a parameter one is seeking to measure or estimate, while in logic, it is the derivation of conclusions from given information or premises by any acceptable form of reasoning [12]. Inferences are classified as either *deductive* or *inductive*. They differ in that in the deductive case the truth of the premises guarantees the truth of the conclusion, whereas in the inductive case the truth of the premise lends support to the conclusion without giving absolute assurance. Bayesian inference is a member of the inductive class and uses probability theory as the framework for induction. More specific it uses Bayes' formula to compute the conditional posterior probability that the hypothesis is true, given what was observed. The calculation involves the unconditional prior probability that the hypothesis is true, as well as the conditional probabilities of getting what was observed given the hypothesis and given the alternative. With this as background, reasoning can be said to be the process of drawing inferences appropriate to the situation [13].

For network structures as considered here, the ability and computational feasibility of *propagation* is an important property. Propagation can be seen as the process of spreading something abroad or into new regions. This chapter deals with the question of how to define the network reasoning mechanisms for updating the system's belief in a failure. Using the terms just introduced, the goal is to define how to propagate evidences of occurred events in order to reason about oil production failures.

The question has already been asked in the literature how to translate the probabilities obtained for an event in advice to an operator [6]. It is thus important that the specialist environment of oil production agrees in advance how e.g. "70% probability of a production anomalie in 2 hours if event E occurs" should be interpreted. Motivated by the fact that even a simple formalism can be valuable if recommandations are understandable to the user, this chapter first discusses how reasoning can be done in Association Rules Networks. ARN does not have a clear formalism of reasoning, and the suggestion is to use correlation between nodes expressed by using the concept of shortest-path in graphs. Next, fundamentals of reasoning in Bayesian networks are introduced, and importance is put on reasoning with temporal information. Finally, a comparison of the suggested reasoning schemes ends this chapter.

4.1 Reasoning in ARN

Reasoning in ARN has been minimally discussed in the literature. Starting with blank pages, it is possible to think of several ways of reasoning, using e.g. other uncertainty formalisms than that of probability theory, or even measures initially not related to uncertainty at all.

4.1.1 Fundamentals of ARN reasoning

Chawla et al. [15] suggest that an ARN goal node can be explained by the hyperpaths leading to it, thus they see the reasoning process as the problem of finding optimal[1] hyperpaths in cycle free B-graphs. If V_{max} is the set of all maximum level nodes in an ARN, then for each $v \in V_{max}$ let be P_v be the set of all hyperpaths from v to the goal node. The authors define two cost measures based on confidence for each hyperpath p:

1. $Weight(p) = \sum\limits_{e_i \in p} log_2(conf(e_i))$
2. $Info(p) = -\sum\limits_{e_i \in p} conf(e_i) log_2(conf(e_i))$

It is suggested that $Weight(p)$ can be interpreted as the strength of the correlation between the source and the goal node, and $Info(p)$ can be interpreted as the total information gain along the path from the source to the goal node. The optimal path in P_v using one of the two measures can be seen as the best explanation for dependence of the goal node on v.

The general problem of finding optimal hyperpaths in hypergraphs has been studied by Ausiello et al. [9]. An algorithm of time complexity $O(|H| + nlogn)$ is reported, where $|H|$, the size of the hypergraph, is $\sum\limits_{e_i \in E} (|T(e_i)| + 1)$. In general, running times for directed graph shortest-path algorithms are lower than those of hypergraphs. There exists a broad range of shortest-path algorithms for such graphs, many which are deduced by e.g. [19]. The algorithms vary with the purpose and constraints on the graph. A well known class of problem is the *single-source shortest-path problem*: Given a graph $G = (V, E)$, find the shortest path from a given *source* node $s \in V$ to each node $v \in V$. Algorithms solving this problem can also solve other problems like the *single-destination shortest-path problem*: Find a shortest path to a given *destination* node t from each node v. This can be reduced to a single-source problem by reversing the direction of each edge in the graph.

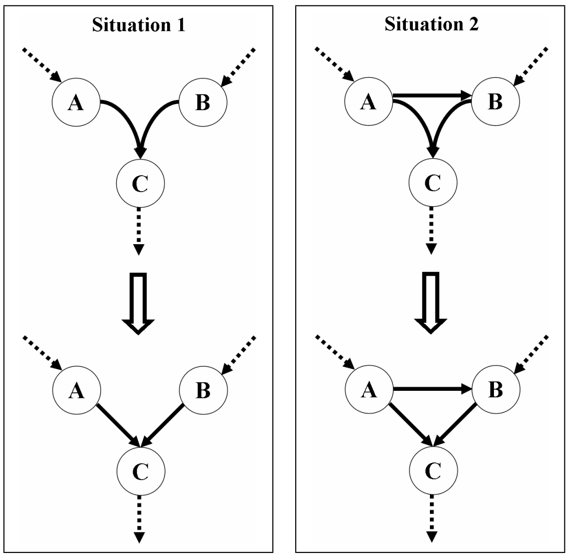
If the graph considered is a directed acyclic graph (DAG), an algorithm called $Dag - Shortest - Paths$ can be used. It has a total running-time of $\Theta(V + E)$, something which is linear in the size of an adjacency-list representation of the graph [19]. This favourable running-time is achieved by using the fact that the nodes of a DAG can be topological sorted. A topological sort of a DAG, $G = (V, E)$, is a linear ordering of all its nodes such that if G contains an edge (u, v), then u appears before v in the ordering. Another benefit of DAGs is that shortest paths are always well defined since even if there are negative-weight edges, no negative-weight cycles can exists. This makes less constraints on the set of possible measures for optimal paths.

[1]The term *optimal path* is used together with *shortest path* since a short path will provide information about the optimal explanation of correlation between two nodes.

4.1.2 Transforming an ARN to a DAG

Both the linear running time of finding the shortest path and the handling of negative-weight edges motivates for a transformation of an hyperhraph-based ARN to a DAG. The transformation of a B-graph to a DAG must therefore be considered. Taking the network structure itself as a starting point, hyperedges must be replaced by single edges, as for the learning process of Bayesian networks in Chapter 3.3.1. ARN, however, *does* distinguish $\{A \rightarrow C, B \rightarrow C\}$ from $\{A, B \rightarrow C\}$, so care must be taken in order to preserve this information in the DAG. In general, components of a former hyperedge must be labelled with this information in order to put correct weights on the edges.

The scheme for putting weights on components of a former hyperedge can be illustrated by the rule $A, B \rightarrow C$ which initially is modelled with a hyperedge from A and B to C. In general there are two cases that must be considered: When there is no edge between A and B (Situation 1: Figure 4.1a) and when such an edge exists (Situation 2: Figure 4.1b).



(a) Situation 1 (b) Situation 2

Figure 4.1: Illustration of hypergraph transformations.

For both situation 1 and situation 2, there are again four cases that must be considered, i.e. combinations of A and B, in order to set the correct edge weights, as shown in Table 4.1. The table cells should be interpreted as giving the rule from where to retrieve the correct weight. As an example consider a situation where event A has occurred. According to the table the edge from A to C should be given the weight of the rule $A \rightarrow C$ and the edge from B to C should be given the weight of the rule $A, B \rightarrow C$.

As the Table 4.1 shows, the two situations are similar but for one case, that is when none of A and B have occurred and the weight for the edge from B to C is needed. The weight in this situation can be found like this:

1. If the path considered (in the shortest-path algorithm) includes the edge between A and B (note that this edge might be one of an earlier hyperedge including other nodes), the measure for $A, B \rightarrow C$ must be used.

	Edge from A to C		**Edge from B to C**	
Situation / Occurred events	1	2	1	2
1. None	$A \rightarrow C$	$A \rightarrow C$	$B \rightarrow C$	Special situation
2. A	$A \rightarrow C$	$A \rightarrow C$	$A, B \rightarrow C$	$A, B \rightarrow C$
3. B	$A, B \rightarrow C$	$A, B \rightarrow C$	$B \rightarrow C$	$B \rightarrow C$
4. A and B	$A, B \rightarrow C$	$A, B \rightarrow C$	$A, B \rightarrow C$	$A, B \rightarrow C$

Table 4.1: Explanation of transformed hyperedge weights

2. Paths not including the edge from A to B must use the measure for $B \rightarrow C$.

Also note that there will never be a path from A to B not including the direct edge between the two as the notion of node level in the ARN learning algorithm ensures this.

The example described can be generalized to account for hyperedges where $|T(e)| = n, n > 2$. Each node in a former hypernode constellation must keep a list of the other former members, and their status (occurred/not occurred). This list must also be updated with temporarily information of what former member nodes that have been visited during the current shortest-path algorithm pass.

4.1.3 Explaining correlation between nodes

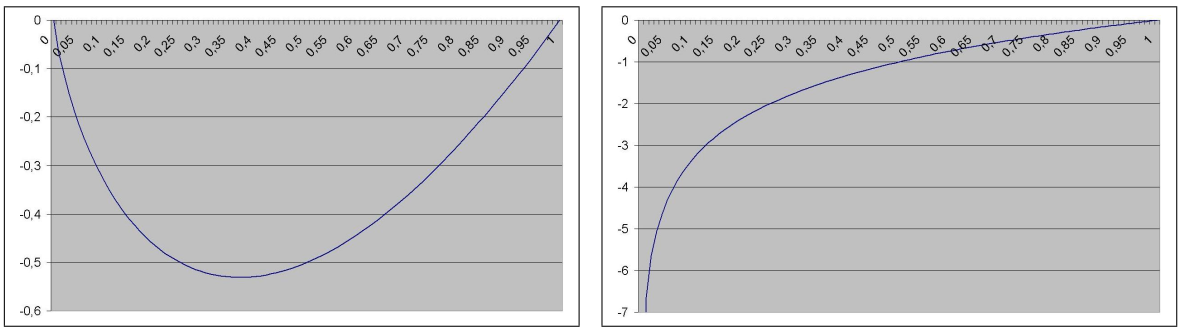
For the specific problem considered, using *single-source shortest-path*, the shortest path from each occurred node should be calculated. Any path including an occurred node that is not the source of the path is redundant and can be left out. Another way of using the information encoded by the ARN is to turn the *single-source shortest-path problem* to a *single-destination shortest-path problem*. By doing this it is possible to find every node whose shortest path to the goal node conforms to a pre-defined level of correlation. The algorithm should be stopped when the level of correlation is diminished, or it comes up against a node that has occurred. In the same way the temporal information can be used, stopping the shortest-path algorithm when the path exceeds a pre-defined amount of time from the root node. It is important that the principles of Chapter 4.1.2 are carefully considered when turning the edge directions in order to execute *single-destination shortest-path*. In fact, Table 4.1 can be used as it is, taking the special situation into account. A check must be performed if the path includes the edge from B to A whether it also includes the edge from C to B (remember that the edges are turned).

The edge weights in ARNs do not have to be probabilistic measures. As long as the path measure gives some sort of explanation of the correlation between the source and the goal node, the measure itself is subordinate. As previously mentioned $Weight(p)$ and $Info(p)$ have been suggested as alternative measures [15]. The latter, being the total information gain along the path from the source to the goal node, is not suited for the domain at hand. Since information gain is the change in entropy, the uncertainty associated with a random variable, from a prior state to a state that takes some information as given, probabilities on both ends of the scale (0 to 1) are treated more or less similarly. In short, the reason for this relates to the amount of information a random variable provides, and in such a context low probabilities can give as much information as a high ones. This is illustrated by Figure 4.2a that shows the graph of the function $f(x) = x \cdot log_2(x)$. It should be clear from this that the shortest-path concept presented here needs a monotonically increasing/decreasing function in order to unambiguously represent the amount of belief in a future oil production anomaly.

$Weight(p)$ is previously stated as representing the strength of correlation between the source and the goal node. Taking a deeper look into the formula it sums the logarithm of the edge weights, in this case the confidence of the rules, which is the same as multiplying the weights and then scaling the answer by taking the logarithm of it. As an example consider a path, p, consisting of the two edges $e_1 = 0.8$ and $e_2 = 0.9$:

$$Weight(p) = log_2(0.8) + log_2(0.9) = -0.47 = log_2(0.8 \cdot 0.9)$$

By multiplying the confidence of the rules, an assumption about rule independence is made. This is due to the definition of statistical independence that states that any collection of events is mutually independent if, and only if, for any finite subset $A_1, \ldots, A_n$ of the collection it is true that $Pr(A_1 \cap \cdots \cap A_n) = Pr(A_1) \cdots Pr(A_n)$. Hence $Weight(p)$ is a log-scaled probabilistic measure assuming rule independence. In proportion to the non-scaled value, log-scaling leads to a greater punishment of lower values. Figure 4.2b shows the graph of the function $f(x) = log_2(x)$. Note that $Weight$ cannot be used straightforward since it, in terms of shortest path, credits the lowest correlation values. This can be fixed by using $Weight(p) = \sum_{e_i \in p} -log_2(conf(e_i))$ instead. From now on $Weight$ should be interpreted this way.



(a) Graph of the function $f(x) = x \cdot log_2(x)$.

(b) Graph of the function $f(x) = log_2(x)$.

Figure 4.2: Illustrations of the weight functions.

As mentioned the edge weights in ARNs do not have to be probabilistic measures. Christiansen [17] discusses a range of interestingness measures for rules, and concludes that support, confidence, lift and J-measure [55] constitute a complement set of measures needed to clean a rule set. J-measure is an information-theoretic measure that combines a bias toward more frequently occurring rules (the antecedent probability), with the degree of surprise in going from a prior probability (the consequent probability) to a posterior probability (the probability of the consequent after observing the antecedent). It is here important to distinguish the interestingness measures used for rule cleaning as well as removing level cycles when learning (see Algorithm 1) from the one used as correlation measure, since not every measure can be combined in a meaningful way. This is the case for J-measure as the antecedent probability is used directly in the calculation of it. To see the inconvenience of this property, consider the simple network of two rules, $A \rightarrow B$ and $B \rightarrow C$. If A has occurred, the correlation between A and B can be found directly by applying the selected rule weight. In order to find the correlation between A and C, the measured correlation between A and B is used as the system's belief in B, and this should be combined with the weight of the last rule in order to find the desired correlation. By using J-measure, this would be meaningless since it contains the marginal probability of B. The same argument hold for A; the event has for sure occurred, so the marginal probability does not matter.

The use of several interestingness measures to clean rules is justified in that they complement each other. Take for example J-measure; it is a symmetric measure that credits both probable and improbable rules, but by only selecting rules with a lift value greater than 1, it becomes asymmetric with respect to positive correlations. Lift also strengthens the belief on confidence as a provider of information contained in a rule as it can be used to prune rules explaining negative correlations. Recall the assumption made that input rules are relevant and interesting. Having a rule set with these properties, one must find the measure that best explains correlation between nodes. Of the four measures suggested by [17], confidence has already been mentioned, and it will be further discussed in Chapter 4.3. It is, however, important to realize that the ARN reasoning mechanism is flexible when it comes to edge weights as long as the best correlation between two nodes can be explained by a shortest path.

4.1.4 Reasoning with temporal information in ARN

To the author's knowledge there has been no research on representing temporal information in ARNs. The temporal information is useful for two purposes: (1) It can be used to update the edge weights, and (2) it provides information to the operator. Since temporal information can alter the edge weights a pass must be made over the network before running the optimal-path algorithm in order to set the correct values. In order to describe this process, the notion of an event's *living time* must be defined. The life analogy is also used for rules.

Definition 7. *An event's living time is the time from when it first occurs to it must be considered without influence. When an event occurs, it is brought to life, and when its living time is up, it dies (or gets killed).*

The following list explains how living times for events and rules can be found:

- The width of the sliding window used for rule mining defines an upper limit for an event's living time, so it is chosen as a measure.
- A rule's living time is the maximum reaction time which is given as a part of the rule, as explained in Chapter 3.1.1. At runtime, a countdown must be initiated with the occurrence of the last antecedent element of a rule. When the number of antecedent elements is one, this is without complications. For composite rules, however, this could lead to a situation where an antecedent event dies before the rule does. A composite rule is pruned and not given as input to a learning process if it does not provide extra information with respect to the corresponding single rules. In other words, the antecedent elements of such rules can together be presumed to have a major influence on the consequent (negative or positive) compared to the effect of each of them alone. Hence it is reasonable to utilize composite rules, even though one or more of the antecedent elements are dead for a period of time.

Tables 4.2a and 4.2b sum up the runtime temporal behaviour of events and rules.

ARN reasoning is a blank page with respect to the extent of research on, and applications of, Bayesian networks. It is built on the principle of simplicity and consequently it lacks expressiveness. Thus, Chapter 4.2 explores reasoning in Bayesian networks which can be viewed as offering expressiveness at the expense of complexity.

Incident	Action
Event occurs	Time-stamp event
A living event re-occurs	Re-stamp event
Event exceeds sliding window width	Kill it

(a) Utilization of event temporal information

Incident	Action
All antecedent events are alive	Time-stamp rule
An antecedent event dies	Do nothing
An antecedent event gets re-stamped before the rule dies	Re-stamp rule
Rule time exceeds given living time	Kill it

(b) Utilization of rule temporal information

Table 4.2: Reasoning with temporal information in ARNs.

4.2 Reasoning in Bayesian networks

Bayesian network reasoning is the process of propagating effects of occurring events to the rest of the network. In Bayesian networks, effects are spread regardless of the direction of the directed edges between nodes. Below, fundamentals of Bayesian network reasoning are described, followed by an explanation of the consequences of introducing time. Special importance is attached to the TBNE reasoning mechanisms.

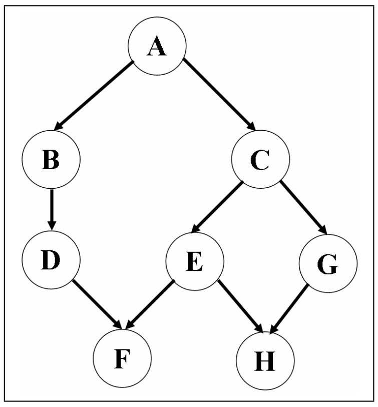
4.2.1 Fundamentals of Bayesian network reasoning

In general, evaluating Bayesian networks is NP-hard [31], hence requiring the need for efficient inference algorithms. Inference can be done by sampling (using e.g. Monte Carlo techniques - see for instance [26] for an introduction) or by using heuristic-based algorithms for exact inference. If the variables considered are continuous (or a mixture of discrete and continuous) sampling techniques are the only choice. In the problem at hand, methods for exact inference can be used. A summary of the process, based on the elaborate review given by Jensen and Nielsen [31], is given next. Note that the process described is not visible to the user, and that the extensiveness of the method reflects the NP-complete nature of Bayesian networks.

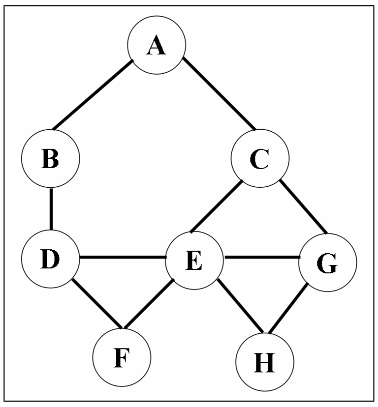
Exact inference is based on a structure called *junction tree*. This structure is compiled offline, i.e. in the construction phase, using *moralization* and *triangulation*. Moralization is the process of "marrying" nodes with one or more common children. In other words, an edge is drawn between any two nodes in the directed graph having an edge pointing to the same node. Considering Figure 4.3a it is easy to see that there are two node-pairs that must be married, namely {D, E} and {E, G}. The moralization step ends with the removal of edge directions, with the result of Figure 4.3b.

An undirected graph is triangulated if, and only if, every cycle of length four or greater contains an edge that connects two non-adjacent nodes in the cycle. Inference complexity is exponential in clique-sizes, so emphasis should be made to minimize the clique-sizes of the network. A *clique* is defined as a *complete set* (all nodes are pairwise linked) that is not a subset of another complete set. Triangulations with a minimal maximum clique-size is NP-hard so heuristics must be used. Taking Figure 4.3b as a starting point, the cycle that must be altered is {A, B, D, E, C}. Adding an edge between B and C leaves the cycle {B, D, E, C}. To complete the triangulation an edge between D and C is added, as shown in Figure

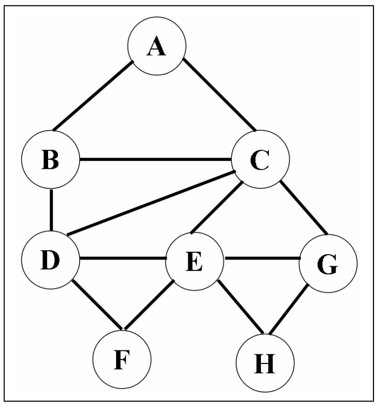
4.3c. Note that this is not a unique solution (an edge between B and E could have been added in the last step).



(a) Bayesian network



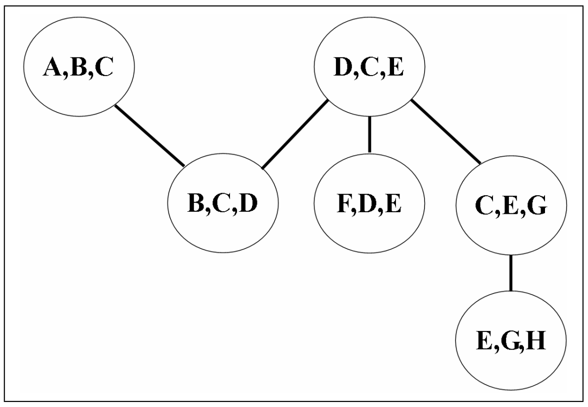
(b) Moralized graph



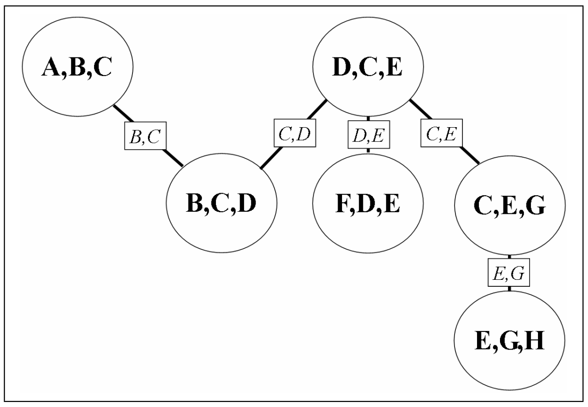
(c) Triangulated graph

Figure 4.3: Illustration of the triangulation process

The triangulated graph has the property that a *join tree* can be built from it. A join tree is built by organizing the set of cliques from the undirected graph in a tree. In a join tree, for any pair of nodes V, W all nodes on the path between V and W contain the intersection $V \cap W$. Consider the join tree in Figure 4.4a. On the path between the nodes {A,B,C} and {C,E,G} we find the intersection variable C.



(a) Join tree



(b) Junction tree

Figure 4.4: The join tree and junction tree for the triangulated graph in Figure 4.3c

Finally, each edge in the junction tree is labelled with the intersection of the adjacent cliques. These labels are called separator sets, or separators, and the resulting structure forms a junction tree. In this structure, nodes can pass messages to calculate the marginal distributions, hence it is making exact inference possible. Figure 4.4b shows the junction tree for the triangulated graph in Figure 4.3. The circles represents cliques while the squares represents separators.

This explains the underlying structure used when building a system for exact inference in Bayesian networks. Several implementations of the junction tree propagation algorithm exists, and an example of probability updating is given in Chapter 2.2.2.

4.2.2 Reasoning with temporal information in Bayesian networks

In Chapter 3.3.2 the temporal representation TBNE is introduced. An important property of this representation is that it makes it possible to reason using the same algorithms as classic Bayesian networks. As mentioned, the temporal intervals of each node are relative to its parents. In addition, an initial event is used as a temporal reference for the other events. The actual time interval of occurrence for each event is determined by a three-step process [7]:

1. **Event detection and time interval definition**
   The time of occurrence for the initial event, t_{init}, is utilized as a temporal reference for the network. This initial node can be either a root node, an intermediate node or a leaf node. In the first case, the actual value of the node can be determined since root nodes are instantaneous events. In the two other cases, the value of the variable cannot be determined because the event could be associated to any time interval for the state. A second observation has to be made in order to determine the interval. When the next event is detected, its time of occurrence, $t_{posterior}$, is utilized for defining the time interval associated with the real time occurrence function, $\alpha = |t_{init} - t_{posterior}|$. The value of α is used to set the time interval of the child node considering the parent node as the initial event. This step is applied recursively to subsequent events.

2. **Propagation of the evidences**
   Having obtained the value of a node (time interval and associated state), the effect of this value is propagated through the network.

3. **Determination of the past and future events**
   With the posterior probabilities it is possible to estimate the potentially past and future events based on the probability distribution of each temporal node.

In order to deal with situations where there are not enough information (e.g. only one event is observed, which corresponds to an intermediate node), [7] suggests using *scenarios*. The node is instantiated to all intervals corresponding to the observed state, and the posterior probabilities of the other nodes are obtained for each scenario. When other events occur, the scenarios are evaluated accordingly to the new information. As previously argued, time semantics should not complicate the network in such a way that standard propagation techniques cannot be used. Even though the concept of TBNE fulfils this requirement, one should be aware of the complexity of introducing scenarios. By using scenarios several parallel networks will exist until enough information is available.

Also note that restricting the number of states to true/false, does not in general restrict the number of state changes defined per node. In fact, the definition of the number of time intervals and their duration for each node is free (multiple granularity) and can be see as a trade off between the complexity and the accuracy needed for depicting the knowledge of the temporal domain. In Chapter 3.1.1 the possibility of deriving statistical distributions for time between events is mentioned. If desirable this could be combined with the option of multiple granularity in order to give more accurate temporal information, but as mentioned this is a trade-off between complexity and the need for such information.

4.3 Comparing the suggested reasoning schemes

TBNE/Bayesian networks and ARN are two different concepts representing the trade-off between complexity and expressiveness. In Bayesian networks each node has associated with it a conditional probability table that quantifies the effects that the parents have on the node. Taking the graph as a whole, the conditional probabilities and the structure can be used to determine the marginal probability or likelihood of each node holding one of its states. Changing one of these marginal probabilities, the effect is propagated throughout the network, updating the other probabilities. This powerful, but complex process contrasts the one of ARNs that uses a simple approach where effects propagate in one direction only. The ARN reasoning process does not produce answers of exact probability, but offers flexibility in the way correlation between nodes are measured, and runs in linear time.

As an example consider the difference of the simple approach of assuming independence, as for the $Weight$-measure, and the comprehensive mathematical model used by Bayesian networks. The example consists of two rules:

- $A \rightarrow B$ with confidence = 0.8
- $B \rightarrow C$ with confidence = 0.9

Assuming that event A has occurred, $Weight(A \mapsto B \mapsto C) = 0.47$, as shown in Chapter 4.1.3 (with the sign reversed). Without the log-scaling this value becomes 0.72. In the Bayesian network case the calculations becomes as follows:

$$\begin{aligned}
Pr(C|A) &= Pr(C, B|A) + Pr(C, \neg B|A) \\
&= \frac{Pr(A, B, C)}{Pr(A)} + \frac{Pr(A, \neg B, C)}{Pr(A)} \\
&= \frac{Pr(A)Pr(B|A)Pr(C|A, B)}{Pr(A)} + \frac{Pr(A)Pr(\neg B|A)Pr(C|A, \neg B)}{Pr(A)} \\
&= Pr(B|A)P(C|B) + Pr(\neg B|A)Pr(C|\neg B) \\
&= 0.8 \cdot 0.9 + Pr(\neg B|A)Pr(C|\neg B) \\
&= 0.72 + 0.2 \cdot Pr(C|\neg B)
\end{aligned}$$

In other words, the Bayesian calculation takes into consideration the probability of C when B is true *and* when B is false, while the simple ARN approach underestimates the value by only considering the positive example. It is worth noting that the value of $Pr(\neg B|A)$, corresponding to the rule $A \rightarrow \neg B$, can be expected to be very low if the network in general consists of rules with high values of confidence. The underestimate, however, becomes even greater in cases where a node has more than one parent since the number of possible state combinations grows exponential.

Nevertheless, it should not be concluded that ARN does not provide valuable information to the user. By using the ARN approach, it is the different values within the ARN that will be compared. Recall that the question of how to translate probabilities in advice to the operator does not have a clear answer. Hence, it is important that the system's belief of a production failure is expressed in an understandable, unambiguous manner, regardless of the underlying calculations. The important task is the one of translating various calculated results to user-friendly terms, and this is an iterative, expert-driven task that must be executed with care for each domain considered.

Table 4.3 summarizes the reasoning mechanisms of ARN and TBNE.

Property	**ARN**	**TBNE**
Complexity	Linear-time shortest path algorithms can be used by transforming the hypergraph to a DAG	NP-hard, heuristics must be used
Effect of occurring events	A shorter path to the goal node may be found	Propagated throughout the network
Expressiveness	Node correlations can be calculated using several measures, but the interpretation of the answers must be carefully investigated	The marginal distribution for each node is available

Table 4.3: Comparing the reasoning mechanisms of ARN and TBNE

An example that illustrates the discussed differences between ARB and TBNE, is given next. The fictitious rule set found in Chapter 3.2.1 is used as a starting point, but rule weights and time intervals are added. Maximum rule time is set to 60 minutes, and the full specifications can be found in Appendix A.2. Reasoning in ARN will be done by the *single-source shortest-path*-principle, using the $Weight$-function with confidence as input. Screen shots from Bayesian network reasoning are created by using Netica [43]. Note that Netica does not support the full concept of TBNE, but the example is chosen carefully in order to avoid ambiguities. Finally, the goal node, representing an anomalous situation, is the node labelled X.

As previously shown, the network built from the rule set will be like the one of Figure 4.5.

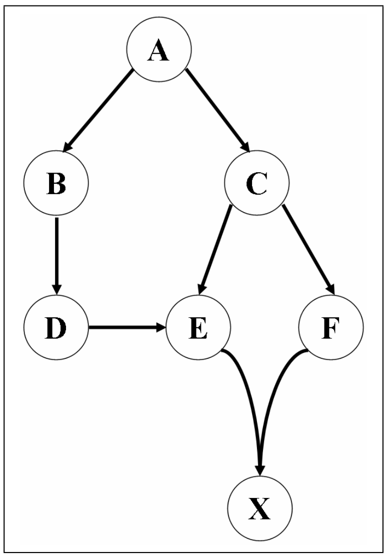
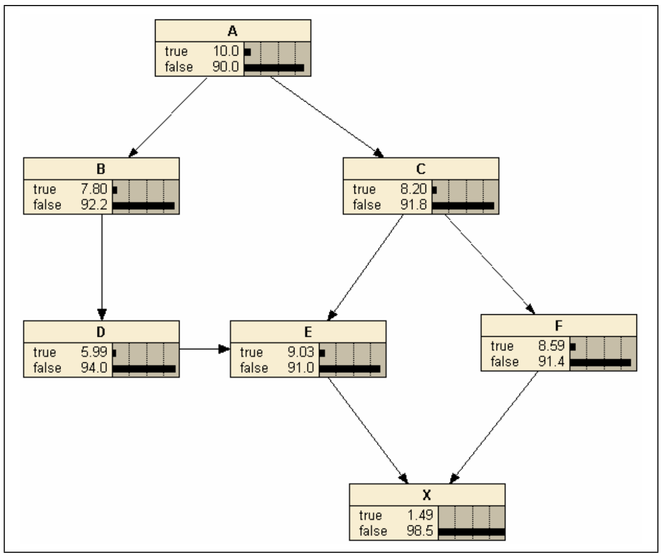
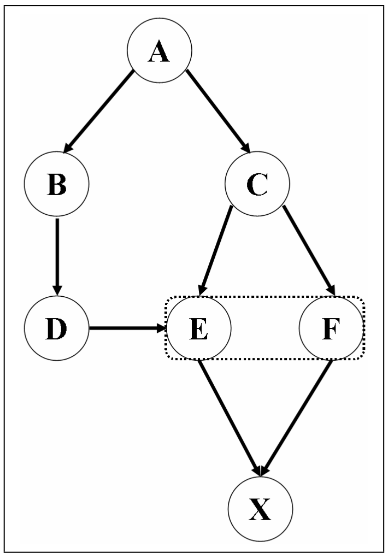


Figure 4.5: Generic network learned from a fictitious rule set

The generic network is next transformed to both a TBNE and a DAG-ARN, as shown in Figure 4.6 (temporal intervals are omitted in order to increase readability). The dotted line surrounding nodes E and F symbolises that they have previously constituted the tail of a hyperedge. For the sake of simplicity, only TBNE-figures will be used for illustration from now on.



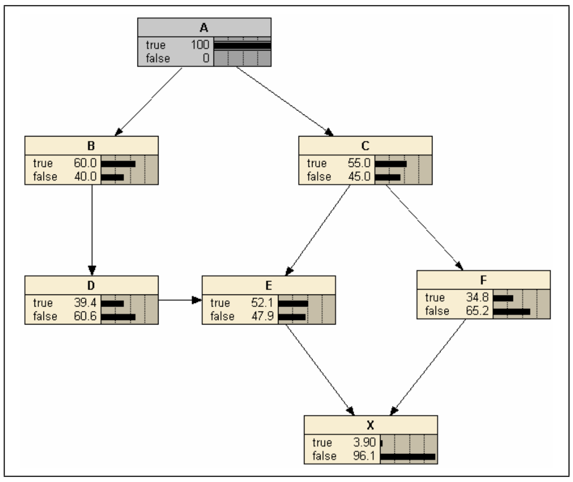
(a) Initial TBNE



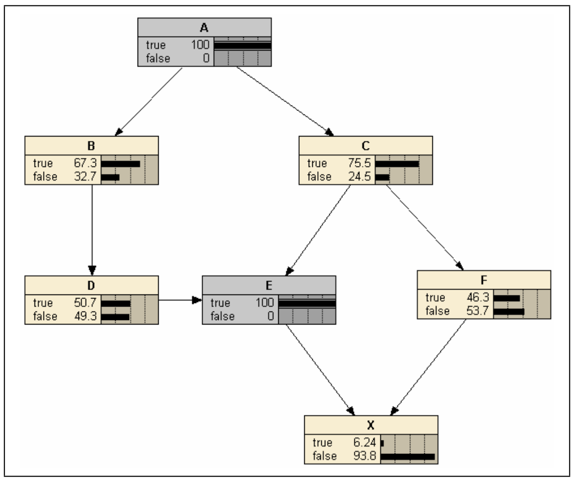
(b) Initial ARN

Figure 4.6: TBNE and ARN after transformation

Assume now that event A occurs. TBNE uses a Bayesian network reasoning mechanism to update the marginal probabilities of the other nodes, and the result can be seen in Figure 4.7a. It shows among others that the probability of the goal node, X, is calculated to be 3.90%. As for ARN, consider first the two paths leading to E: $Weight(A \mapsto B \mapsto D \mapsto E) = -log_2(0.6) - log_2(0.65) - log_2(0.6) = 2.095$ and similar $Weight(A \mapsto C \mapsto E) = 1.484$. In other words, the latter represents the part of the shortest path going from A to X via E. Finally, $Weight(A \mapsto C \mapsto E \mapsto X) = 6.543$ and $Weight(A \mapsto C \mapsto F \mapsto X) = 7.243$, so the path via E is in fact the shortest path from A to X in the network. If this value should cause an alarm, the path can be used as an explanation of the correlation between the source and the goal node.



(a) TBNE after the occurrence of A



(b) TBNE after the occurrence of E

Figure 4.7: TBNE after the occurrence of A and E respectively

After only 30 minutes, event E occurs, and the TBNE updates the marginal probability of X to 6.24%, as seen in Figure 4.7. Note also the increase of belief in C that has been raised from 55.0% to 75.5%, and should become a candidate for careful monitoring. According to the temporal intervals, this probability will hold for another 25 minutes. Taking a look at ARN, there are two paths to be considered: $Weight(E \mapsto X) = 5.059$ and $Weight(A \mapsto C \mapsto F\&E \mapsto X) = 4.921$. The including of E in the latter path conforms to the scheme of Table 4.1. This also explains why the path from A just becomes the shortest one, since the probability of X given E *and* F is much higher than if just one of them has occurred. Finally, the temporal information presented to the user is *"If C within 25 minutes ⇒ F within 55 minutes after C ⇒ X within 50 minutes after F"*.

This example illustrates some of the differences between TBNE and ARN. Consider first the explanation of the goal node. TBNE calculates its marginal probability and a visual inspection of the network is needed in order to discover the influencing nodes. The ARN method explains its findings by stating the path from the source to the goal node. In the example above it would not be unnatural to explain several paths, at least after the occurrence of E. Implicit this also highlights another difference since in TBNE the choice of node(s) to monitor does not affect the reasoning. By using ARN, the reasoning mechanism needs to know in advance the node to which the shortest path shall be calculated. This is important since there might be situations where other nodes than the one representing an anomalous situation should be monitored (and alarms possibly given). Finally, the previously discussed underestimation of ARN reasoning is illustrated since the path $A \mapsto C \mapsto E \mapsto X$ does not take into account the increased probability of F (and hence underestimates the belief in X).

Even though the example is fictitious, it illustrates a likely property of the domain in that the probability of the target node is small under certain conditions. Truly interesting and novel relations for the target event, with a high probability, are assumed to be found when the number of antecedent elements increases [17]. If this is the case, a careful consideration should be done before using ARN due to the underestimation property.

Chapter 5

Discussion and conclusion

The combining of local and global data mining techniques should be done stepping carefully between the two research traditions. On the other hand, an important realisation is that only a single successful prediction of an oil production anomaly will save ConocoPhillips from a major loss of income. Thus, details from a discussion between the two traditions are subordinate to suggestions of how a system that will help the operators to do their job even better can be made. Looking back at the motivation and the problem description it is clear that a pragmatic view on the case is reasonable.

This chapter first considers a short-term solution that can be implemented and tested immediately at ConocoPhillips, followed by a discussion taking a long-term view on the issue. Emphasis is put on discussing consequences of choices made. Next this is also done for the overall process of network learning and reasoning, considering the three-step process of data utilization in the oil production domain as well. Finally, this master thesis' contributions to the field of expert systems are discussed, before a conclusion is given at the end of the chapter.

5.1 Suggested solutions

For ConocoPhillips it is valuable to receive suggestions on how oil production data can be utilized in a short-term view. Taking such a solution as a starting point, a discussion on its strengths and weaknesses is also desirable. This knowledge should then be used as foundation for a more advanced solution, taking a long-term view on the issue.

5.1.1 A simple solution

The preceding chapters have shown that the concept of ARN constitutes a simple, but effective solution to the problem. By transforming the network to a directed acyclic graph (DAG), linear running times can be achieved for the reasoning algorithms. The $Dag - Shortest - Paths$ algorithm is well known, and the only obstacle to overcome is to implement the dynamic edge-weight scheme of Table 4.1. In order to explain correlations between nodes, the $Weight$-measure is suggested, using confidence as input. Reasoning should be done calculating the shortest path from every occurring node, giving a warning if the value exceeds a pre-defined threshold. In what follows, important properties of this solution are discussed.

Complexity
The transformation of a B-graph to a DAG is previously justified by the reduced running time of the shortest-path algorithms. This is, however, not a one-sided issue, as the transformation results in a non-trivial edge-weight scheme. If the resulting network is small, the reduced running time is ignorable. Normally, rule mining processes produce a vast number of rules, but the rule cleaning operation preceding the network learning might reduce this number considerably. It is therefore advisable to chose the type of network after the structure learning, evaluating each case for itself.

Independent of the type of network, the temporal behaviour of events and rules adds complexity to the network. Recall the summary of living times as given in Table 4.2. The updating scheme itself is trivial, but it requires frequently traversals of the network in order to perform the updates.

Correlations
The $Weight$-measure is previously shown to be an underestimate of the probability since it assumes independence between variables. Its main advantage is the simple calculation, using a measure that can be easily calculated during rule mining. It biases high confidence levels, and the log-scaling makes the total score to drop dramatically when total confidence falls beneath 0.3 (see Figure 4.2b). For domains where small values of confidence can be expected, the suitability of the method must be thoroughly considered. If this is mainly the case for rule where the target node acts as a consequence, as mentioned in Chapter 4.3, a solution is to monitor nodes preceding the target.

In principle any interestingness measure can be used, as long as the results are interpretable. The ARN shortest-path method requires communication with domain experts in order to decide what range of values that should cause a warning (or multiple ranges for graded warnings).

Type of reasoning
A drawback of the suggested reasoning method, i.e. calculating the shortest path from every occurring node, is the lack of utilization of temporal information. The operator might be interested in rules where the maximum time from the goal node is limited to e.g. 8 hours. A question can also be asked whether it is necessary to calculate the shortest path every time a node occurs. Both of these concerns can be dealt with using the principle of *single-destination shortest-path*. In fact, by setting thresholds in advance, a network can be learnt including only those nodes meeting the operator's requirements. By doing this, if the network is small enough, it is possible to reduce the online monitoring to be that of a visual inspection of the network only.

The main advantage of the suggested reasoning method is the information completeness with respect to the recently suggested alternatives since they restrict the number of nodes in the network, hence, leaving out information of possibly novel correlations. Even though it can be argued that the temporal information could be better utilized than in the suggested solution, the information is presented with each rule constituting a shortest path. Thus, it is up to the operator to take advantage of this information. This is the core of expert systems: They shall help users doing a better job, not replace them.

To sum up ARN shortest-path provides a simple and intuitive solution to the failure prediction problem. Temporal information is easily encoded and the shortest-path algorithms are considered well known. Independent of the network size, fast reasoning mechanisms exists. The simplicity is also the main drawback of the solution since the answers provided are inaccurate. Use of domain experts is required in order to obtain the range of values that should cause a warning.

5.1.2 An advanced solution

TBNE represents a more comprehensive solution built upon the framework of Bayesian networks. The combination of encoding temporal information and at the same time keeping the standard propagation mechanisms of such networks is what makes TBNE appealing. Transforming the ARN network structure to a TBNE is easily done, but requires communication with the rule mining process in order to retrieve missing values for the various CPTs. As for the simple solution, important properties of this one is discussed next.

Complexity
As described earlier Bayesian networks are complex structures, considering both learning and reasoning. Since the learning problem is dealt with, it is mainly the reasoning process that adds complexity to the solution considered. The NP-complete nature of Bayesian network reasoning limits the practicable size, but it is the encoding of temporal information that should be given most attention. There exists a range of commercial available implementations of various reasoning algorithms for Bayesian networks, but to the author's knowledge this is not the case for TBNEs. In other words, a solution based on the concept of TBNE must either be built from scratch or adapted to an available solution.

Correlations
In Bayesian networks in general, the marginal probability of every node is calculated based on the information provided by the other nodes. For TBNE in particular, the temporal information adds important knowledge of a node's state. Also recall that effects are spread regardless of the direction of the directed edges between nodes, hence it is not appropriate to speak of node correlations as previously done for ARNs. Even though probability is a known term to most people, it is difficult to exactly explain the difference between 70% and 80% probability of an event. Domain experts should be consulted before deciding the thresholds for warnings.

In the Bayesian network introduction of Chapter 3.3 it was stated that emphasis should be put on avoiding to include conditional independences which do not hold in the real world. By learning networks from rules, it is difficult to prove whether such independences exist or not. On the other hand, the easy learning of temporal intervals from rules might outnumber this disadvantage. One way of seeing this problem is to view it in the light of the weighing between global and local methods, and take it into account when evaluating the overall solution. It is also possible to let a domain expert remove errors from the resulting networks. Note that by using an expert the solution is biased towards his knowledge, and for the complex domain at hand this might not be preferable.

To sum up TBNE, as a temporal extension of Bayesian networks, provides a sustained framework for probabilistic reasoning. It is a powerful tool where marginal probabilities can be found for every node of interest. Its complexity is also what makes it difficult to implement, especially taking the temporal properties into account. By using rules as a basis for such networks, it is also important to be aware of unwanted conditional independences that might be encoded.

A comparison of the most important properties of the two solutions is given in Table 5.1.

	Simple, ARN-base	**Advanced, TBNE-based**
Complexity	Simple	Major, especially due to encoding of temporal information
Correlations	$Weight$ is used as distance measure. The distance from every occurred node to the goal node is calculated	Calculates the marginal probability for every node. A visual inspection of the network can be done in order to discover which nodes that influence the target node most
Variants	Several implementation variants can be considered	If a finer granularity of the temporal intervals is supported by the rule mining process, this can be implemented in the TBNE
Role of domain expert	Interpret correlation measure in order to set warning thresholds	Interpret probabilities in order to set warning thresholds, maybe prune network in order to remove unwanted independences
Main drawbacks	Inaccurate correlation measures that can be difficult to interpret	Complexity, might encode independences that does not hold in the real world
Main advantages	Simple to implement, fast reasoning mechanism	Accurate calculations, the framework of Bayesian networks is well known

Table 5.1: Properties of the suggested solutions

5.2 Evaluating the overall solution

Having considered two specific solutions to the learning and reasoning problem, this section provides a more general discussion of the subjects, also considering the recommended three-step process of data utilization in the oil production domain.

Considering the assumptions on learning input from Chapter 3.1, one of them should be discussed more thoroughly. Take the parallel antecedent assumption: It is likely that some information is lost with respect to the alternative of using serial antecedents, but the amount is unknown. A disadvantage of restricting the antecedent to be serial is its impact on possible reasoning networks as e.g. Bayesian networks cannot be used without greater modifications of the concept. The strict requirement of events occurring in a given order might quickly lead to a complex situation as an event can be a part of several rules, playing a different role in each of them. Hence, the choice of using parallel antecedents is a trade-off between expressiveness and complexity.

Another important subject is time, and the question must be asked what consequences leaving temporal information out of the process will have. As already seen the encoding and use of temporal information adds complexity to the process, especially when it comes to the concept of Bayesian networks. The main argument for using temporal information is expressiveness, i.e. the additional information it gives the operator. At the same time it has been argued that the operator does not need an exact distribution of the time, but a coarse description of what the system believe will happen. A solution could have been to drop temporal information from the learning and reasoning process, and provide some general properties of the rules instead. This solution depends on the nature of the rules. If most rules lies within the same temporal interval, the solution could be as good as the alternative, but if the intervals vary a lot, dropping the temporal information might cause dropping extremely important information as well.

Argumentation can also be given for the opposite view that the temporal information suggested is too coarse-grained. TBNE supports the encoding of several temporal intervals for a node, so more accurate statements of temporal relations can be expressed. The consequence of a finer granularity is increased complexity of the network. In short, the encoding of temporal information is a choice concerning the user's need, the availability of the information (from the rule mining process) and a balance between expressiveness and complexity.

Looking back to the guidelines for data utilization (Chapter 1.3.3) and the particular problem case of emulsification in the low pressure separator of process 2/4 J (Chapter 1.3.2), the framework of Mollestad [40, 41] outputs a set of tags that are relevant to the target, and constitutes the input data for the next step of discovering dependencies between events. The work of Christiansen [17] shows that it is possible to mine restricted association rules and clean the (possible huge) set of rules in order to give as input to the learning process only the most interesting and relevant ones. With the results of this thesis it is therefore likely that the overall solution will add valuable information to the task of pre-warning anomalous situations in oil production processes, and in particular to the problem case of process 2/4 J.

Nevertheless, a question must be asked whether other solutions can be suggested with the knowledge obtained from the previously discussions of network learning and reasoning. As written in Chapter 3.3.1 Bayesian networks can be learned by combining user knowledge and statistical data. In order to carry out such a process the data should be stored in database form. The output from the first step of the overall solution can be seen as a continuous time line of events, hence it must be transformed to fulfil the database requirement. A solution to this might be to use the notion of sliding windows, i.e. to let each sliding window be a database row. Assuming this method can be used, the next issue to consider is temporal information. The founders of TBNE obtained time intervals for each node based on knowledge about the modelled process combined with data from a simulator [7]. This method limits the size of domain to be modelled and also violates the thoughts of discovering new and novel relations in the data. As mentioned in Chapter 3.3.3 Bayesian networks and association rules have been applied together in iterative processes to overcome the latter issue. An extension of these thoughts in order to comprise temporal information should not be deemed impossible.

This short discussion shows that other approaches than the one suggested is possible. It does, however, require a different view on how to solve the problem. The three step of the overall solution used as a model are dependent on each other and they are developed from a holistic point of view. In general, the main advantage of this model is its intuitive appeal and the possibility of combining large amounts of data and automated processes with expert knowledge. For the step of knowledge utilization, as discussed in this thesis, the user gets a framework for combining rules in order to be pre-warned of anomalous events. It is up to the user to select target node, node(s) to monitor, and the various alarm thresholds. Cooperation between knowledge engineers and domain experts in the process of setting up the system is crucial in order to achieve success. Finally, there is no such thing as *the* solution, i.e. every case is different an must be treated thereafter.

5.3 Contributions

This thesis contributes to the field of expert systems by combining thoughts from both rule-based systems and probabilistic expert systems. More specific, its most novel suggestions are:

- Building a network structure from rules by further developing the concept of ARN. Most importantly are:
  - The improvement of the learning algorithm described in the literature.
  - Transformation of an ARN hypergraph to a DAG.
  - Reasoning in the transformed ARN, including temporal information.
- The combination of association rules and Bayesian networks, methods which origin from two different research traditions. The rules are used both for building the network structure and for setting the values of the CPTs. Further, it suggests using the temporal information encoded by the rules to extend the solution to the concept of TBNE.

5.4 Conclusion

In this thesis methods have been derived for combining event patterns in time series, called restricted association rules, in order to warn about future anomalies in oil production processes. It conforms to the third step of knowledge utilization in a guideline for data utilization in the oil production domain developed by Mollestad [40, 41]. Two parts of the knowledge utilization step have been explored: Network learning and network reasoning.

Network learning is the problem of how to learn a network structure from restricted association rules. A concept called Association Rules Network (ARN) [14] is proposed as a framework for learning a network structure from a set of rules. ARN is a hyper-graphical model for representing rules whose consequents are singletons. Weak points in the description of the ARN learning algorithm reported in the literature are uncovered, and based on this an improved version is presented.

Network reasoning is the process of propagating evidences of occurred events to the rest of the network in order to update the system's belief of oil production failures. Reasoning in ARN is done by calculating correlation between nodes based on the shortest-path principle, so to reduce the complexity of reasoning the hypergraph-based ARN is transformed to a DAG. By doing this, linear running time algorithms can be used. It is proven that by computing the shortest path by the $Weight$-function when confidence is used as edge weights, the ARN reasoning mechanism underestimates the real probability values. However, as even exact measures of probability is not easily translated to the operators, the results of the ARN reasoning process stands for itself. Thorough investigation of what values that should raise an alarm must be done by domain experts in advance of operational use. Due to its simplicity and intuitive appeal the ARN method is suggested as a solution, taking a short-term view on the issue.

Bayesian networks are one of the most well known concepts of probabilistic reasoning. Motivated by the shortcoming of ARN the relations between association rules and Bayesian networks have been investigated. It is pointed out that there do not exists any common applications of such networks based solely on mined association rules, and that they in fact are examples of methods from different research traditions within the field of data mining. Another obstacle is the lack of direct support for representing temporal dependencies in Bayesian networks. Based on this, a concept named Temporal Bayesian Network of Events (TBNE) [7] is introduced. TBNE enjoys the properties of Bayesian network reasoning while at the same time representing temporal information. This is achieved by altering the node semantics such that a node's conditional probability table reflects the probability that the event will occur within some time of its parents.

The problem of learning TBNE from a set of restricted association rules is suggested solved by taking the hypergraph-based ARN as a starting point and transforming it to a valid Bayesian network structure. The temporal intervals and corresponding probabilities for each node can partly be obtained from the rules, but communication with the rule mining process is required in order to retrieve missing values. Where ARN stands for simplicity, TBNE offers expressiveness at the expense of complexity, hence TBNE is suggested implemented in the longer term.

In a broader perspective the thesis contributes to the field of expert systems by combining thoughts from both rule-based systems and probabilistic expert systems. It is important to stress the role of the expert, not just in operational mode, but also when setting up the system. Both the network structure and interpretation of reasoning results should be developed in cooperation between knowledge engineers and domain experts.

This thesis has shown that it is theoretically feasible to combine restricted association rules in order to create a network structure for reasoning. It is concluded that the final choice of solution must be based on a careful consideration of the trade-off between complexity and expressiveness. For further work a natural continuation is testing the concepts of ARN and TBNE with real data. In particular, an interesting goal of research is to explore exactly how the node correlations provided by the ARN reasoning mechanism should be interpreted in a given domain. Also, a study should be done to measure the size of the underestimate of using confidence, with respect to the exact values. Further, a closer integration with the rule mining process should be carried out in order to evaluate the size and nature of the rule set given as input to the learning task. Finally, an interesting study would have been to compare the performance of the overall solution used as a model for this thesis with a solution based solely on Bayesian learning methods.

Bibliography

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993. 2.3, 2.3, 4

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann, 12–15 1994. 2.3

[3] ConocoPhillips - 2006 Annual Report. http://www.conocophillips.com, 30 May 2007. 1.2

[4] G. Arroyo-Figueroa, Y. Alvarez, and L. Sucar. SEDRET - an intelligent system for the diagnosis and prediction of events in power plants. *Expert Systems with Applications*, 18:75–86, 2000. 3.3.2, 3.3.2

[5] G. Arroyo-Figueroa and L. Sucar. A temporal bayesian network for diagnosis and prediction. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 13–20, San Francisco, CA, 1999. Morgan Kaufmann. 3.3.2

[6] G. Arroyo-Figueroa and L. Sucar. Temporal bayesian network of events for fault diagnosis and prediction in thermal power plants. In J. M. Agosta, O. Kipersztok, K. B. Laskey, K. W. Przytula, and I. Rish, editors, *Proceedings of the First Bayesian Applications Modeling Workshop*, 2003. `http://www.intel.com/research/events/UAI03_workshop`. 4

[7] G. Arroyo-Figueroa and L. E. Sucar. Temporal bayesian network of events for diagnosis and prediction in dynamic domains. *Applied Intelligence*, 23(2):77–86, 2005. 3.3.2, 3.3.2, 4.2.2, 4.2.2, 5.2, 5.4

[8] G. Arroyo-Figueroa, L. E. Sucar, and A. Villavicencio. Probabilistic temporal reasoning and its application to fossil power plant operation. *Expert Systems with Applications*, 15:317–324, 1998. 3.3.2

[9] G. Ausiello, G. F. Italiano, and U. Nanni. Hypergraph traversal revisited: Cost measures and dynamic algorithms. In *MFCS '98: Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science*, pages 1–16, London, UK, 1998. Springer-Verlag. 4.1.1

[10] J. Bowes, E. Neufeld, J. E. Greer, and J. Cooke. A comparison of association rule discovery and bayesian network causal inference algorithms to discover relationships in discrete data. In *AI '00: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 326–336, London, UK, 2000. Springer-Verlag. 3.3.3

[11] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264. ACM Press, 1997. 2.3

[12] Inference defined by Encyclopedia Britannica. Encyclopedia Britannica Online: `http://www.britannica.com/eb/article-9042389`. Last visited 25 April 2007. 4

[13] Reasoning defined by Encyclopedia Britannica in the article "Artificial Intelligence". Encyclopedia Britannica Online: `http://www.britannica.com/eb/article-219080`. Last visited 25 April 2007. 4

[14] S. Chawla, B. Arunasalam, and J. Davis. Mining open source software (OSS) data using association rules network. Technical Report TR535, School of IT, University of Sidney, Sidney, NSW, Australia, 2003. 3.2, 3.2, 3.2.1, 5.4

[15] S. Chawla, J. Davis, and G. Pandey. On local pruning of association rules using directed hypergraphs. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 832, Washington, DC, USA, 2004. IEEE Computer Society. 3.2, 3.2, 3.2, 3.2.1, 3.2.2, 4.1.1, 4.1.3

[16] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning bayesian networks from data: an information-theory based approach. *Artificial Intelligence*, 137(1-2):43–90, 2002. 3.3.3

[17] J. Christiansen. A Framework for Discovering Interesting Rules from Event Sequences with the purpose of Pre-warning Oil Production Problems. Master's thesis, Norwegian University of Science and Technology, 2007. 1.3.3, 3.1, 3.1.1, 3.1.1, 4.1.3, 4.3, 5.2

[18] J. Christiansen and P. K. Helland. Mining time series in order to predict loss in oil production. Technical report, Norwegian University of Science and Technology, 2006. 1.3.3, 3.1.1, 3.3.2, 3.3.3

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001. 4.1.1

[20] R. G. Cowell, S. L. Lauritzen, A. P. David, and D. J. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1999. 2.2, 2.2.1

[21] J. de Jongh and ConocoPhillips. Business and Company Background COP, internal document. 2006. 1, 1.2, 1.3

[22] R. Durrett. *Probability: theory and examples*. Duxbury Press, second edition, 1996. 2.1

[23] C. Fauré, S. Delprat, J.-F. Boulicaut, and A. Mille. Iterative bayesian network implementation by using annotated association rules. In S. Staab and V. Svatek, editors, *Proceedings of the 15th International Conference on Knowledge Engineering and Knowledge Management EKAW'06*, October 2006. 3.3.3

[24] F. H. Fossan. "Masteroppgave - domenespørsmål". Personal email. Fredrik Høymer Fossan is a Maintenance Optimisation Engineer at ConocoPhilips. The mail is dated 11 May 2007. 3.1.1

[25] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2-3):177–201, 1993. 2.4

[26] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo Methods in Practice*. Chapman and Hall, London, 1996. 4.2.1

[27] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, pages 85–96, 1994. 3.3.1, 3.3.1

[28] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, 2000. 2.3, 2.3

[29] J. Hollmén, J. K. Seppänen, and H. Mannila. Mixture models and frequent sets: Combining global and local methods for 0–1 data. In *SIAM International Conference on Data Mining (SDM'03)*, San Fransisco, may 2003. 3.3.3

[30] S. Jaroszewicz and D. A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 178–186, New York, NY, USA, 2004. ACM Press. 3.3.3

[31] F. V. Jensen and T. D. Nielsen. *Bayesian Networks and Decision Graphs II*. Forthcoming. 3.3, 4.2.1

[32] K. Karimi and H. J. Hamilton. Finding temporal relations: Causal bayesian networks vs. c4.5. In *ISMIS '00: Proceedings of the 12th International Symposium on Foundations of Intelligent Systems*, pages 266–273, London, UK, 2000. Springer-Verlag. 3.3.3

[33] U. Kjaerulff. dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting, Special Issue on Probability Forecasting*, 11:89–111, 1995. 3.3.2

[34] M. Klemettinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. I. Verkamo. Finding interesting rules from large sets of discovered association rules. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, pages 401–407. ACM Press, 1994. 2.3

[35] E. Lamma, F. Riguzzi, A. Stambazzi, and S. Storari. Improving the sla algorithm using association rules. In *AI*IA*, pages 165–175, 2003. 3.3.3

[36] D. V. Lindley. *Making decisions*. John Wiley and Sons Ltd, second edition, 1985. 2.1

[37] B. Liu, W. Hsu, S. Chen, and Y. Ma. Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems*, 15(5):47–55, 2000. 2.3

[38] H. Mannila. Local and global methods in data mining: Basic techniques and open problems. In *ICALP '02: Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 57–68, London, UK, 2002. Springer-Verlag. 3.3.3

[39] V. Mihajlovic and M. Petkovic. Dynamic bayesian networks: A state of the art. Technical Report TR-CTIT-01-34, Enschede, October 2001. 3.3.2

[40] T. Mollestad. ConocoPhillips (COP) - Event modelling. Forthcoming. 1.3.3, 5.2, 5.4

[41] T. Mollestad. Warning process problems using an event-based interpretation to time series. Forthcoming. 1.3.3, 5.2, 5.4

[42] M. N. Moreno, S. Segrera, and V. F. López. Association rules: Problems, solutions and new applications. In *III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005*, pages 317–323, 2005. 2.3

[43] Norsys software corp., Netica. http://www.norsys.com, 12 March 2007. 3, 4.3

[44] K.-C. Ng and B. Abramson. Uncertainty management in expert systems. *IEEE Expert: Intelligent Systems and Their Applications*, 5(2):29–48, 1990. 2.1

[45] A. Onisko, P. J. F. Lucas, and M. J. Druzdzel. Comparison of rule-based and bayesian network approaches in medical diagnostic systems. In *AIME '01: Proceedings of the 8th Conference on AI in Medicine in Europe*, pages 283–292, London, UK, 2001. Springer-Verlag. 2.2

[46] S. Parsons and A. Hunter. A review of uncertainty handling formalisms. In *Applications of Uncertainty Formalisms*, pages 8–37, London, UK, 1998. Springer-Verlag. 2.1

[47] D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1409–1421, 2003. 3.3.3

[48] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 133–136, 1982. 2.2.2

[49] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988. 2.1

[50] G. Piatetsky-Shapiro. Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases*, pages 229–248. AAAI/MIT Press, 1991. 2.3

[51] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 3.3.2

[52] J. Roberto J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. *Data Min. Knowl. Discov.*, 4(2-3):217–240, 2000. 3.1.2

[53] E. H. Shortliffe and B. G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23(3–4):351–379, 1975. 2.2.1

[54] C. Silverstein, S. Brin, R. Motwani, and J. D. Ullman. Scalable techniques for mining causal structures. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 594–605, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. 3.3.3

[55] P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992. 4.1.3

[56] P. Tse and J. Liu. Mining associated implication networks: Computational intermarket analysis. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 689, Washington, DC, USA, 2002. IEEE Computer Society. 3.3.3

[57] J. D. Young and E. Santos. Introduction to temporal bayesian networks. In M. Gasser, editor, *1996 Midwest Artificial Intelligence and Cognitive Science Conference*, April 1996. 3.3.2

Appendix A

Data and calculations

The appendix contains data and calculations explicitly referred to in the text.

A.1 Marginalization example of Chapter 2.2.2

In Chapter 2.2.2 it is stated that, given the variables and conditional probabilities in the given example, the marginal probability for having a headache is 0.0958. In other words

$$Pr(Headache = yes) = \sum_{Fl}\sum_{BV}\sum_{Fe} Pr(Headache = yes, Fever, Flu, Bad\ ventilation) = 0.0958.$$

Calculations of marginal probabilities in Bayesian networks are a complex task, and the full calculation of $Pr(Headache = yes)$ is here shown to give the reader not familiar with the concept a deeper understanding of it. Note that for the sake of simplicity, the variables are named the following way: Headache (H), Fever (Fe), Flu (Fl) and Bad ventilation (BV).

$$\begin{aligned}
Pr(H = yes) &= \sum_{Fl}\sum_{BV}\sum_{Fe} Pr(H = yes, Fe, Fl, BV) \\
&= \sum_{Fl}\sum_{BV}\sum_{Fe} Pr(BV) \cdot Pr(Fl|BV) \cdot Pr(Fe|Fl, BV) \cdot Pr(H = yes|Fe, Fl, BV) \\
&= \sum_{Fl}\sum_{BV}\sum_{Fe} Pr(BV) \cdot Pr(Fl) \cdot Pr(Fe|Fl) \cdot Pr(H = yes|Fl, BV)
\end{aligned}$$

In other words, for all combinations of states for Fl, BV and Fe, the product $Pr(BV) \cdot Pr(Fl) \cdot Pr(Fe|Fl) \cdot Pr(H = yes|Fl, BV)$ must be calculated. The sum of these products is the desired value, i.e. the marginal probability of having a headache. The final calculations are shown in Table A.1.

Fl	**BV**	**Fe**	**Product**	**Sum**
Yes	Yes	Yes	$0.01 \cdot 0.15 \cdot 0.80 \cdot 0.99$	0.00119
Yes	Yes	No	$0.01 \cdot 0.15 \cdot 0.20 \cdot 0.99$	0.00030
Yes	No	Yes	$0.01 \cdot 0.85 \cdot 0.80 \cdot 0.90$	0.00612
Yes	No	No	$0.01 \cdot 0.85 \cdot 0.20 \cdot 0.90$	0.00153
No	Yes	Yes	$0.99 \cdot 0.15 \cdot 0.05 \cdot 0.30$	0.00223
No	Yes	No	$0.99 \cdot 0.15 \cdot 0.95 \cdot 0.30$	0.04232
No	No	Yes	$0.99 \cdot 0.85 \cdot 0.05 \cdot 0.05$	0.00210
No	No	No	$0.99 \cdot 0.85 \cdot 0.95 \cdot 0.05$	0.03997
				Total sum: 0.0958

Table A.1: Calculation of the marginal probability $Pr(Headache = yes)$

A.2 Data for the example of Chapter 4.3

The following tables specify the CPTs for the example of Chapter 4.3. The time intervals are written using TBNE notation. In a TBNE, each temporal node is defined as a set of ordered pairs (σ, τ) where σ is the state or value variable and τ is the time interval associated with each state variable attribute. The conditional probability distribution for each node is defined as the probability of each ordered pair (σ_i, τ_i) given the ordered pair of its parents (σ_j, τ_j).

True	**False**
10%	90%

Table A.2: Node A

A	**True, [5–55]**	**False, [0–60]**
True	60%	40%
False	2%	98%

Table A.3: Node B

A	**True, [5–55]**	**False, [0–60]**
True	55%	45%
False	3%	97%

Table A.4: Node C

B	**True, [5–45]**	**False, [0–60]**
True, [5–55]	65%	35%
False, [0–60]	1%	99%

Table A.5: Node D

C	**D**	**True, [5–40]**	**False, [0–60]**
True, [5–55]	True, [5–45]	90%	10%
True, [5–55]	False, [0–60]	65%	35%
False, [0–60]	True, [5–45]	60%	40%
False, [0–60]	False, [0–60]	1%	99%

Table A.6: Node E

C	**True, [5–45]**	**False, [0–60]**
True, [5–55]	60%	40%
False, [0–60]	4%	96%

Table A.7: Node F

E	**F**	**True, [5–50]**	**False, [0–60]**
True, [5–40]	True, [5–45]	10%	90%
True, [5–40]	False, [0–60]	3%	97%
False, [0–60]	True, [5–45]	2%	98%
False, [0–60]	False, [0–60]	1%	99%

Table A.8: Node X