GEORG-AUGUST-UNIVERSITÄT
GÖTTINGEN

ISSN 1612-6793

Nr. ZFI-MSC-2017-04

Master's Thesis

submitted in partial fulfilment of the
requirements for the course "Applied Computer Science"

Identification of Correlations and Root Causes in Event Sequences

Marc-André Zöller

Institute of Computer Science

Bachelor's and Master's Theses
of the Center for Computational Sciences
at the Georg-August-Universität Göttingen

15. February 2017

Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

☎ +49 (551) 39-172000

FAX +49 (551) 39-14403

✉ office@informatik.uni-goettingen.de

www.informatik.uni-goettingen.de

First Supervisor:	Jun.-Prof. Dr.-Ing. Marcus Baum
Second Supervisor:	PD Dr.-Ing. Marco Huber

I hereby declare that I have written this thesis independently without any help from others and without the use of documents or aids other than those stated. I have mentioned all used sources and cited them correctly according to established academic citation rules.

Göttingen, 15. February 2017

Abstract

Event correlation is the task to detect dependencies between different events in a sequence, e.g. a log file. Many domains utilise this knowledge for maintenance or prediction of future events. Even though it is desirable to detect those temporal dependencies automatically, no generic approach exists so far resulting in either domain specific solutions or manual work.

In this work a new data-driven, generic algorithm for event correlation is presented. It uses a fast preliminary test statistic to determine promising event pairs. Next, the precise distribution of the temporal lag between those pairs is calculated. Therefore, two new methods are introduced. The first one—called ICE–calculates temporal distributions with a precision equal to state-of-the-art approaches but with a fractional amount of runtime making it optimal for fast, rough estimates. The second algorithm LpMatcher computes way better estimates with equal or less runtime than state-of-the-art approaches. Finally, a method is developed to automatically generate Bayesian network allowing complex inference and reasoning.

Using synthetic sequences and two real-world data sets from different domains, the effectiveness of this approach and advantages over existing algorithms is shown. Complex and diverse distributions are successful identified and transformed into Bayesian networks.

Contents

List of Figures

List of Tables

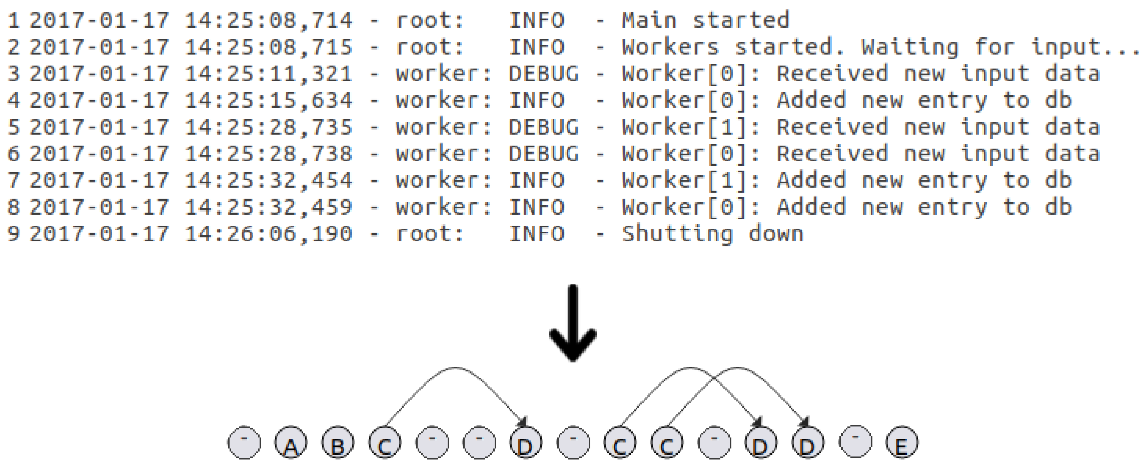
List of Abbreviations

DAG	Directed Acyclic Graph	36
EM	Expectation-Maximisation	8
ICE	Iterative Closest Events	13
ICP	Iterative Closest Point	13
ILP	Integer Linear Programming	28
KDE	Kernel Density Estimator	10
LP	Linear Programming	29
MISE	Mean Integrated Squared Error	11
NP	Non-Deterministic, Polynomial Time	31
PCC	Pearson Correlation Coefficient	14
RANSAC	Random Sample Consensus	21
SAT	Boolean Satisfiability Problem	31

Chapter 1

Introduction

Nearly every modern computer system provides information about its status. This status information—usually collected by logging—is often aggregated in a monitoring system for further inspection and error handling. Usually, this data can be interpreted as a sequence of events. An event is an abstract representation of something that has happened—for example a button was clicked, an exception occurred in a program or a hardware component changed its state. In the most basic case an event has to have some kind of label saying what this event represents and a time stamp when the event has occurred. In addition, events are often enriched with arbitrary supplementary data like a message, component description, input data, etc. A series of events is aggregated in a sequence $\boldsymbol{S}$. Figure 1.1 shows how a log file can be interpreted as a sequence of generic events.



```
2017-01-17 14:25:08,714 - root:   INFO  - Main started
2017-01-17 14:25:08,715 - root:   INFO  - Workers started. Waiting for input...
2017-01-17 14:25:11,321 - worker: DEBUG - Worker[0]: Received new input data
2017-01-17 14:25:15,634 - worker: INFO  - Worker[0]: Added new entry to db
2017-01-17 14:25:28,735 - worker: DEBUG - Worker[1]: Received new input data
2017-01-17 14:25:28,738 - worker: DEBUG - Worker[0]: Received new input data
2017-01-17 14:25:32,454 - worker: INFO  - Worker[1]: Added new entry to db
2017-01-17 14:25:32,459 - worker: INFO  - Worker[0]: Added new entry to db
2017-01-17 14:26:06,190 - root:   INFO  - Shutting down
```

Figure 1.1: Example transformation of a log file to a sequence. The actual log entries are replaced by generic labels while preserving the order of all events. Correlated events are connected by arrows.

These events often influence and trigger each other forming a network of dependencies. With a certain complexity of the underlying system, a manual inspection and handling of all monitored events becomes quite time consuming and therefore expensive. A major challenge during the analysis is that a single fault in an application often does not lead to a single event in monitoring. Instead, a fault—normally referred to as *root cause*—often propagates through a system leading to consecutive errors depending on the first fault. This flood of events makes determination of the actual problem complicated and time consuming.

Starting from the 1980s people developed methods to automatically aggregate and preprocess monitored events to reduce the manual work or completely eliminate it. Yet even after over 30 years of research, no generic framework for event correlation has been developed. Instead, many solutions for specific application areas have been proposed. Nearly all of these solutions incorporate domain specific knowledge making it hard, if not impossible to transfer the solution to other areas.

The purpose of this work is to develop a new data-driven approach for event correlation with minimal knowledge of the underlying system and domain. Data-driven in this context means that only a generic classification of an event (e.g. a numeration or labelling) and its occurrence time stamp are known. It is reasonable to assume that these two pieces of information are always given. Based on this fundamental information, a generic framework for event correlation is introduced.

Chapter 2

Basics

This chapter provides a general introduction to the topic *event correlation*. First, the problem is described and the terminology for this work is defined. Next, a short historical overview of approaches for event correlation is given. Especially, the *lagEM* algorithm by Zeng et al. [1] is explained in detail as it is used as reference for performance evaluation in the later chapters. Finally, the fundamental concepts of hypothesis testing and kernel density estimation are explained.

2.1 Problem Description

Event correlation is the task to detect correlations between different event types within a sequence. By examining the dependencies between the events, a directed graph can be calculated. In this graph, each node represents exactly one event type. A directed connection between two nodes means that event type A triggers B. Using this knowledge it is possible to say which event types will follow after the current event predicting future states of a system. Another application is using the graph to find the root cause of a given event. Both abilities are highly desired in many domains like system monitoring, stock market, intrusion detection systems, etc. for *predictive maintenance* and *root cause analysis* [1], [2]. See Figure 2.1 for an example dependency graph.

Mathematically the problem can be formulated as follows: Let a sequence $\boldsymbol{S}$ containing a set of event types E be given. Desired are all event pairs E_i, $E_j \in E$ such that E_i and E_j are not independent of each other. Those event pairs are called *correlated*.

Definition 1 (Independence)
Two event types E_i, E_j are independent if and only if

$$P(E_i, E_j) = P(E_i) \cdot P(E_j) \tag{2.1}$$

with $P(E_i)$ the probability of E_i and $P(E_i, E_j)$ the joint probability of E_i and E_j. Independence can also be written as $E_i \perp\!\!\!\perp E_j$.

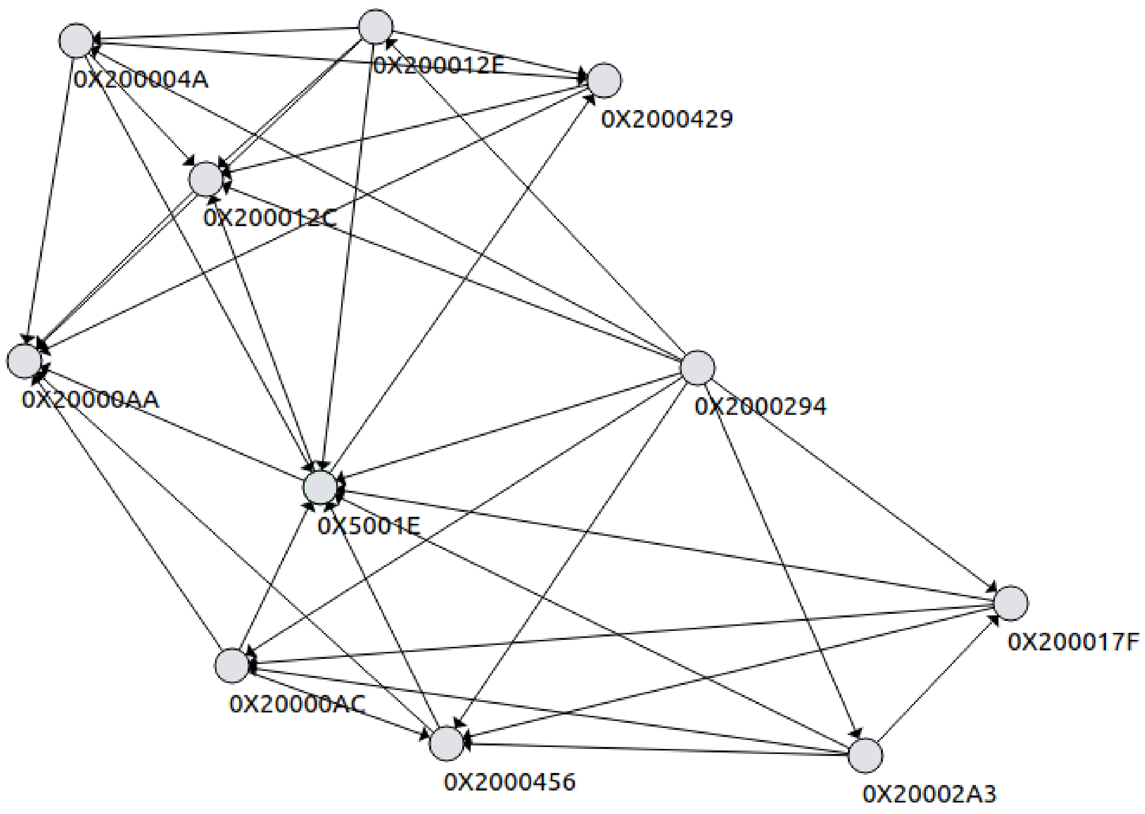


Figure 2.1: Example dependency graph. Each node represents an event type with a hexadecimal label. The arrows show the correlation between two event types.

Even though independence of two events is easy to check, event correlation is difficult as neither $P(E_i)$ nor $P(E_i, E_j)$ are known a priori in general. This task gets even more complicated as two events do not have to appear at the same time even though they are correlated. Often, one event—called the *trigger event*—happens a few seconds or minutes before the correlated event—called the *response event*. For example, assume an application running on a server. The trigger event is a failure of the network card and the response event is an application alerting the lost network connection after some connection retries. The time between the network card failure and the detected network loss is called *temporal lag*.

Definition 2 (Temporal lag)
Let E_i, E_j be two correlated event types with E_i triggering E_j. The event E_j occurs after E_i with a temporal lag

$$P(E_j \mid E_i) = \mathcal{T}_\Delta(E_j \mid E_i) = \mathcal{T}_\Delta(E_j - E_i) \tag{2.2}$$

with $\mathcal{T}_\Delta$ being an arbitrary probability density function. The temporal lag is also called time lag.

For simplicity Table 2.1 contains the complete terminology with a short description. This terminology is going to be introduced throughout this work.

Symbol	Description
$\boldsymbol{S}$	A sequence of arbitrary events.
$\boldsymbol{S}_A$	The sequence of events with type A. Any Latin character can be used as an event type.
$\lvert\boldsymbol{S}\rvert$	The length of the sequence $\boldsymbol{S}$.
E	The set of all event types.
a_i	The ith event of type A. a_i may also refer to the time stamp of the ith event of type A if used in formulas.
d_{ij}	The temporal distance between the ith event of type A and the jth event of type B.
$\mathcal{N}(\mu, \sigma)$	A normal distribution with mean μ and standard deviation σ.
$\mathcal{U}(a, b)$	A uniform distribution between a and b.
$\mathcal{T}_\Delta(E_j \mid E_i)$	The distribution of the time lag between two event types.
$\mathbb{E}(X)$	The expectation value of the random variable X.
$\pi(x)$	A random permutation of the vector x.
$\hat{f}_h(x)$	A kernel density estimator for the vector / sample set x.
$\mathbb{1}\,(x > y)$	An indicator variable. It tests whether a Boolean formula, in this case $x > y$, is true. If that is the case, the indicator variable returns 1, otherwise 0.

Table 2.1: Complete overview of important terminology.

2.2 Existing Approaches

Starting from the early 1980s people investigated event correlation. In the last three decades numerous papers have been published trying to tackle this problem in different ways. This section briefly summarises the most important categories of approaches.

Expert System

The oldest approaches utilise domain experts to create a knowledge database. By asking a domain expert, explicit rules and dependencies between event types are stated. Based on these rules, events are aggregated and correlated [3], [4]. Other approaches extend these ideas by including knowledge about the dependencies of the system components [5]. Even though the basic idea of this approach is quite simple, it is not very universal. First of all, it is not possible to transfer this system to a new domain as all rules have to be recreated. Furthermore, the creation of rules is very time consuming, difficult and error prone for non-trivial systems.

Learning from Historical Data
A second important category utilises previous solutions for similar situations. Often events are aggregated as tickets in a ticket-system. Based on similar tickets in the past, solutions for the new ticket are presented to an operator. Therefore, events can be enriched by manual descriptions and additional information to allow *structural correspondence learning*. Many algorithms have been proposed for this task like clustering using k-nearest neighbours [6] or Bayesian networks [7]. However, all these methods have in common that the correlation between two event types has to be manually detected at least once. This information is preserved and converted into a rule for future events. Consequently, a fully automatic analysis of a new domain is not possible. Furthermore, no knowledge about the actual temporal lag distributions is obtained.

The so far proposed approaches include some domain specific data to find correlations. This may make event correlation for a given problem easier but it prevents a generalisation to other domains. Therefore, a different set of algorithms was proposed that only utilise an event label and a time stamp.

Time Window
The most important generic approach utilises time windows to find frequent event pairs, e.g. [8], [9] and [10]. A time window with a fixed size is shifted over the complete sequence like an n-gram over a text. By counting how often event types occur in the same time window, frequent pairs can be found. Based on these frequent pairs, *episodes* with several correlated event types can be detected. Figure 2.2 visualises this procedure.

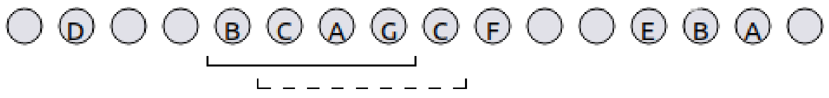


Figure 2.2: Visualisation of time window. The black box defines a time window with length 4. The dashed box shows the next time window.

A major drawback of this approach is that no reasonable way for choosing the window size exists. If the window size is too small, relevant event pairs may be missed. However, if the window size is too large, many false positive correlations will be detected. Furthermore, this method does not allow calculating the actual distribution of the temporal lag between two event types.

lagEM Algorithm

An interesting existing approach is the *lagEM* algorithm proposed by Zeng et al. [1]. This algorithm is used as a performance reference for event correlation and is therefore explained more detailed than the other existing approaches.

The *lagEM* algorithm models the temporal lag between two event types A and B by a normal distribution. By using *Expectation-Maximisation* (EM), the likelihood of the model parameters—namely mean μ and variance σ^2—is maximised. More formally, the equation

$$\hat{\Theta} = \arg\max_{\Theta} P(\Theta \mid \boldsymbol{S}_A, \boldsymbol{S}_B) \tag{2.3}$$

with Θ being the model parameters is solved. By applying Bayes' theorem [11], $P(\Theta \mid \boldsymbol{S}_A, \boldsymbol{S}_B)$ can be calculated as

$$\ln P(\Theta \mid \boldsymbol{S}_A, \boldsymbol{S}_B) = \ln P(\boldsymbol{S}_B \mid \boldsymbol{S}_A, \Theta) + \ln P(\boldsymbol{S}_A) + \ln P(\Theta) - \ln P(\boldsymbol{S}_A, \boldsymbol{S}_B)\,.$$

By eliminating terms not depending on Θ and arguing the prior of Θ is uniformly distributed, Zeng reduced the assignment problem to

$$\begin{aligned}\hat{\Theta} &= \arg\max_{\Theta} P(\boldsymbol{S}_B \mid \boldsymbol{S}_A, \Theta)\\ &= \arg\max_{\Theta} \sum_{j=1}^{n} \ln P(b_j \mid \boldsymbol{S}_A, \Theta)\end{aligned}$$

with the reasonable assumption that all b_j are mutually independent. To calculate $P(b_j \mid \boldsymbol{S}_A, \Theta)$ and model the relationship between b_j and a_i, a dependency variable x_{ij} is introduced. This variable models how likely it is that a_i triggered b_j

$$\begin{aligned}P(b_j \mid \boldsymbol{S}_A, \Theta) &= \prod_{i=1}^{m} P(x_{ij} = 1) \cdot P(b_j \mid a_i, \Theta)\\ &= \prod_{i=1}^{m} \pi_{ij} \cdot \mathcal{N}(b_j - a_i \mid \mu, \sigma^2)\end{aligned}$$

with $\pi_{ij} = P(x_{ij} = 1)$. The above reformulation incorporates the assumption that the time lag is normally distributed. This obtains the final objective

$$(\mu, \sigma^2) = \arg\max_{\mu,\sigma^2} \sum_{j=1}^{n} ln \sum_{i=1}^{m} \pi_{ij} \cdot \mathcal{N}(b_j - a_i \mid \mu, \sigma^2)\,. \tag{2.4}$$

The final objective is solved by an EM algorithm. An EM algorithm starts with a randomly selected model. Next, it calculates alternatingly an assignment of data points based on the current model—called the *expectation step*—and a new model based on the current assignment—called the *maximisation step*. These two steps are repeated until the model converges [12].

For the *lagEM* algorithm, the model initialisation is done by setting $\pi_{ij} = 1/n$ and choosing μ and σ^2 randomly. Next, the expectation step calculates assignment probabilities based on the model of the previous step (indicated by a dash)

$$\pi_{ij} = \frac{\pi'_{ij} \cdot \mathcal{N}(b_j - a_i \mid \mu', \sigma^{2'})}{\sum_{j=1}^{n} \pi'_{ij} \cdot \mathcal{N}(b_j - a_i \mid \mu', \sigma^{2'})} . \tag{2.5}$$

Finally, the maximisation step simply calculates the mean μ and variance σ^2 based on the time lag of all events weighted by their assignment probability.

$$\mu = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_{ij} \cdot (b_j - a_i) \tag{2.6}$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{m} \sum_{j=1}^{n} \pi_{ij} \cdot (b_j - a_i - \mu)^2 \tag{2.7}$$

Using Equations (2.5), (2.6) and (2.7) an EM algorithm can be used to solve the event correlation problem. A detailed performance evaluation of this algorithm is done in Section 4.1.

2.3 Hypothesis Testing

During the process of event correlation a decision has to be made whether two event types are correlated with each other or not. Two event types are considered as correlated if their occurrence time stamps are correlated. To decide if the occurrence time stamps are correlated, a hypothesis test can be used.

A hypothesis test is a well defined procedure to test if a random variable has a specific feature, e.g. a specific expectation value. This specific feature is the hypothesis. To assess the hypothesis, a random experiment is done. Normally, the procedure contains the following steps [13]:

1. **Initial Hypothesis**: At first a null hypothesis H_0 has to be chosen. This null hypothesis—in this case that two event types are not correlated—is going to be tested. In addition, an alternative hypothesis H_1 exists stating the opposite of H_0. The alternative hypothesis is that two event types are not correlated.

$$
\begin{aligned}
H_0 : &\quad P(E_i, E_j) = P(E_i) \cdot P(E_j) \\
H_1 : &\quad P(E_i, E_j) \neq P(E_i) \cdot P(E_j)
\end{aligned}
$$

2. **Test Statistic**: Next, a suited test statistic has to be chosen. This test statistic is utilised to decide whether H_0 is accepted or not. Often Student's t-distribution is used [14]. Section 3.1 introduces the relevant test statistics for this work.

3. **Random Sampling**: By sampling of the random variable, a value t_{obs} is calculated. How exactly t_{obs} is calculated highly depends on the actual test statistic and cannot be generalised. For event correlation, a sequence with both event types represents the random sample.

4. p**-Value**: Finally, based on t_{obs}, the probability of a more-extreme result is calculated as $p_{\text{left}} = P(T \leq -t_{\text{obs}} \mid H_0)$ and $p_{\text{right}} = P(T \geq t_{\text{obs}} \mid H_0)$. The null hypothesis is rejected if [15]

$$2 \min(p_{\text{left}}, p_{\text{right}}) \leq \alpha \tag{2.8}$$

   with a preliminary chosen significance level α. Usually, α is chosen as 0.01 or 0.05.

Using these four steps, a profound statement can be made whether two event types are correlated in a given sequence.

2.4 Kernel Density Estimator

A *Kernel Density Estimator* (KDE) is a non-parametric method to estimate a probability density function of a random variable. This method is used to obtain the unknown distribution based on a random sample set [16].

Definition 3 (Kernel Density Estimator)
Let $(x_1, \ldots, x_n)$ be a sample set of independent and identically distributed values. The KDE for this sample set is defined as

$$\hat{f}_h(x) = \frac{1}{n \cdot h} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.9}$$

with h being the bandwidth and $K(x)$ a kernel.

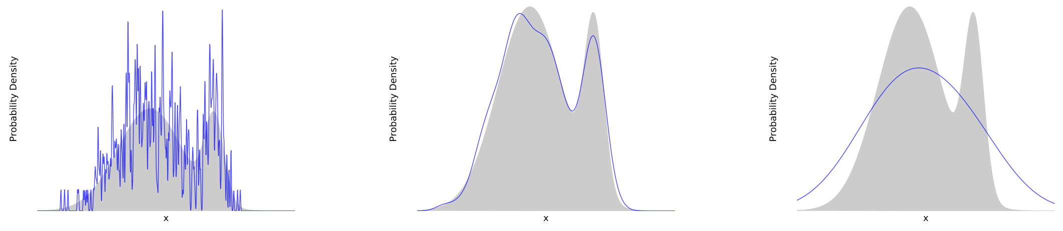
Definition 4 (Kernel)
Let $K : \mathbb{R} \to \mathbb{R}$ be a function. K is called a kernel if it fulfils the constraints

- $\int\limits_{-\infty}^{+\infty} K(x) \ dx \ = \ 1$
- $K(-x) = K(x) \quad \forall x \in \mathbb{R}$
- $K(x) \geq 0 \quad \forall x \in \mathbb{R}$.

These three properties enforce that the kernel results in probability density function with zero mean. Many different kernel types exist and are used. One very commonly used kernel type is the Gaussian kernel [17]

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-0.5x^2\right) . \tag{2.10}$$

Besides the kernel, a bandwidth h has to be chosen. The bandwidth is used as a smoothing parameter. Figure 2.3 visualises the effects of different bandwidths for the same underlying distribution. If the bandwidth is selected too small, the resulting distribution will contain much noise. However, if the bandwidth is selected too large, important features of the underlying distribution may be missed.



(a) Bandwidth $h = 0.01$ (b) Bandwidth $h = 0.2$ (c) Bandwidth $h = 1$

Figure 2.3: Influence of the bandwidth on the estimated distribution. The grey function is a combination of two normal distributions. The blue function is the estimated KDE distribution.

Often it is not possible to provide reasonable values for h, therefore the bandwidth is estimated from the input samples. Simple approaches use an ad hoc method like [18]

$$h = n^{-1/(d+4)} \tag{2.11}$$

with n being the number of samples and d the dimension of each sample point. More sophisticated methods try to minimise the *Mean Integrated Squared Error* (MISE)

$$\mathrm{MISE}(h) = \mathbb{E}\left[\int_{-\infty}^{+\infty} \left(\hat{f}_h(x) - f(x)\right)^2 \, dx\right] \tag{2.12}$$

as this yields an optimal bandwidth [19]. Yet, this equation is not solvable as it depends on the true distribution $f(x)$. Therefore, several methods have been proposed to approximate the MISE, e.g. [20], [21]. In the context of this work, Equation (2.11) is sufficient to estimate the bandwidth.

Chapter 3

Methods

This chapter presents a generic approach to the event correlation problem. Listing 3.1 shows a high-level overview in pseudo code.

```
1    seq = Sequence.load('...')
2    result = []
3    for trigger in seq.eventTypes:
4        for response in seq.eventTypes:
5            correlation = distance(trigger, response)
6            if (correlation < 0.05):
7                distribution = matcher.match(trigger, response)
8                result.add(distribution)
9    network = BayesianNetwork.train(result)
```

Listing 3.1: High-level overview of event correlation algorithm.

This approach investigates correlations between event type pairs. Therefore, all pair of event types have to be analysed (see line 3 and 4). At first, a hypothesis test is done whether both event types are correlated. A metric to rate the correlation between two event types is introduced in Section 3.1. If this test is positive (see line 5 and 6 in Listing 3.1), the actual distribution of the temporal lag is calculated (line 7). For this step, two different algorithms are presented. First, an adapted version of the *Iterative Closest Point* (ICP) algorithm called *Iterative Closest Events* (ICE) is explained in Section 3.2. Next, the problem is solved by formulating a linear minimisation problem called *LpMatcher* as described in Section 3.3. Finally, all calculated pair-wise distributions are transformed into a Bayesian network (see line 9 and Section 3.4.2). This Bayesian network can be used for manual analysis or automatic inference.

3.1 Dependency Measures

In general, calculating the correlation between two event types is a quite time consuming process. If the event sequences get sufficiently large, computing all $O\left(|E|^2\right)$ event pairs requires a significant amount of time. It would be better if only event pairs with a high probability of containing a strong correlation would be examined. This section introduces measures to assess the significance of a correlation without actually computing it. Based on the sample time stamps normalised semimetrics are defined. A value close to 1 means no correlation and a value close to 0 means both sample sets are correlated. Furthermore, all these semimetrics directly allow hypothesis tests (see Section 2.3). Therefore, it is explained how to calculate a p-value for each metric.

Pearson Correlation Coefficient

The most famous measure for correlation of two random variables is the *Pearson Correlation Coefficient* (PCC) [22]. Basically, this method fits a straight line through paired observations and calculates the residuals

$$r = \frac{\sum\limits_{i=1}^{n} (a_i - \mathbb{E}(A))(b_i - \mathbb{E}(B))}{\sqrt{\sum\limits_{i=1}^{n} (a_i - \mathbb{E}(A))^2} \sqrt{\sum\limits_{i=1}^{n} (b_i - \mathbb{E}(B))^2}} \,. \tag{3.1}$$

The coefficient ranges from -1 for negative linear dependency, over 0 meaning no correlation to 1 for linear correlation [23]. As it is only desired to know whether both samples are correlated and not *how* they are correlated, the correlation coefficient is defined as

$$\mathrm{Cor}_{PCC}(\boldsymbol{S}_A, \boldsymbol{S}_B) = 1 - |r| \,. \tag{3.2}$$

To obtain a p-value from the correlation coefficient, first a t-value has to be calculated. Therefore, the correlation coefficient is corrected by the standard error of the correlation coefficient [24]

$$t = \frac{r}{\sqrt{\frac{1-r^2}{n-2}}} \,. \tag{3.3}$$

Using t, the p-value can be directly computed from the Student's t-distribution $f_n(x)$ [22].

$$p = 2 \int\limits_{-\infty}^{-|t|} f_{n-2}(x)\, dx \tag{3.4}$$

PCC has two major drawbacks. As it can only capture linear correlations, a non-linear correlation is missed by this semimetric. Furthermore, both sample sets have to be bijectively paired meaning that the exact correlation has to be computed first. Consequently, this measure is not suited as a dependency measure.

Mutual Information

Mutual information is a fundamental concept from information theory to assess the dependency between two random variables. Informally, mutual information measures the amount of information, e.g. in bits, that one random variable contains about another one. To define mutual information, *entropy* has to be defined first. Entropy is a measure of the information content of a discrete probability distribution introduced by Shannon [25], [26].

Definition 5 (Entropy)
The entropy of a discrete random variable A with a probability mass function $P(A)$ is defined as

$$\mathbb{H}(A) = -\sum_{a \in A} P(a) \log_2 P(a) \, . \tag{3.5}$$

Yet, as the occurrence time stamps and time lag are continuous random variables, the original definition of entropy is not applicable. The concept of *differential entropy* is an extension of entropy to continuous random variables [26].

Definition 6 (Differential Entropy)
The differential entropy $h(A)$ of a continuous random variable with probability density $P(A)$ is defined as

$$h(A) = -\int_{-\infty}^{\infty} P(a) \log_2 P(a) \, da \, . \tag{3.6}$$

In general it is possible to link the differential entropy to discrete entropy by binning the continuous probability distribution into slices with width Δ. This allows a direct transition from differential entropy to entropy and vice versa.

$$h(A) - \log_2 \Delta \xrightarrow{\Delta \to 0} \mathbb{H}(A) \tag{3.7}$$

In contrast to entropy, mutual information does not measure the information of a single random variable but the information of one random variable about another one. Figure 3.1 shows this correlation.

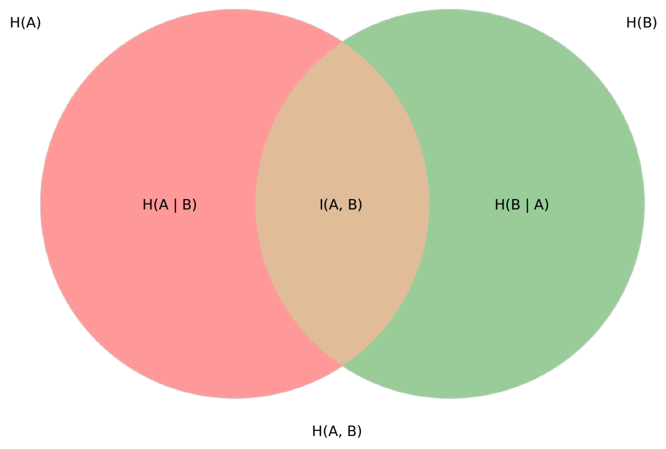


Figure 3.1: Visualisation of the relationship between entropy and mutual information

Definition 7 (Mutual Information)
Let A and B be two random variables with a joint probability density $P(A, B)$ and probability density functions $P(A)$ and $P(B)$. The mutual information of A and B is defined as

$$\begin{aligned}\mathbb{I}(A, B) &= \mathbb{H}(B) - \mathbb{H}(B \mid A) \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(a, b) \log_2\left(\frac{P(a, b)}{P(a) \cdot P(b)}\right) da\, db \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(b \mid a)P(a) \cdot \log_2\left(\frac{P(b \mid a)}{P(b)}\right) da\, db\,.\end{aligned} \tag{3.8}$$

To actually calculate the mutual information $P(A)$, $P(B)$ and $P(B \mid A)$ have to be known. $P(A)$ can be simply computed by counting

$$P(A) = \frac{|\boldsymbol{S}_A|}{|\boldsymbol{S}|}\,. \tag{3.9}$$

It is important to note, that this value is identical for every realisation $a \in A$. $P(B)$ is defined respectively. $P(B \mid A)$ is the temporal lag between A and B and is therefore calculated by ICE in

Equation (3.22) or *LpMatcher* in Equation (3.28). As the temporal lag only depends on the temporal difference between A and B, the calculation can be simplified to a single integral

$$\mathbb{I}(A,B) = \int_{-\infty}^{\infty} \mathcal{T}_\Delta(x) P(A) \cdot \log_2\left(\frac{\mathcal{T}_\Delta(x)}{P(B)}\right) dx\,. \tag{3.10}$$

Finally, this allows calculation of the correlation coefficient

$$\mathrm{Cor}_{MI}(\boldsymbol{S}_A, \boldsymbol{S}_B) = 1 - \frac{\mathbb{I}(A,B)}{\mathbb{H}(A,B)}\,. \tag{3.11}$$

The division by the joint entropy $\mathbb{H}(A,B)$ [26] normalises Cor_{MI}.

To compute a p-value for hypothesis testing, a permutation test is used. Let $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ be bijectively paired sequences of events with b_{a_i} the paired value of a_i. These pairs are used to calculate an empirical mutual information

$$\hat{\mathbb{I}}(\boldsymbol{S}_A, \boldsymbol{S}_B) = \sum_{i=1}^{m} P(b_{a_i} \mid a_i) P(a_i) \cdot \log_2\left(\frac{P(b_{a_i} \mid a_i)}{P(b_{a_i})}\right). \tag{3.12}$$

Next, $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ are permuted to break the pair-wise relation. Based on this permutation, the p-value is calculated as

$$p = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}\left(\hat{\mathbb{I}}(\boldsymbol{S}_A, \boldsymbol{S}_B) > \hat{\mathbb{I}}(\pi(\boldsymbol{S}_A), \pi(\boldsymbol{S}_B))\right) \tag{3.13}$$

with k the number of repetitions of the test. If $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ are truly correlated, the permutation breaks this correlation leading to less mutual information.

Even though mutual information is the preferred measure for correlation in information theory, it has a major drawback. For computation knowledge of the distribution $\mathcal{T}_\Delta(x)$ is required in Equation (3.10). This makes correlation assessment before actual event correlation impossible.

Energy Distance

The *energy distance* is inspired by gravitational potential energy from physics and was recently proposed by Rizzo [27]. This method is an extension of PCC to non-linear dependencies. Like two objects attracting each other in a gravitational field, the samples of two distributions affect each other depending on their "distance". If both sample sets were produced by equivalent distributions their potential energy is zero.

Definition 8 (Energy Distance)
Let F_A and F_B be the cumulative distribution function of A and B. Furthermore, let A' be an independent identically distributed copy of A. The energy distance is defined as

$$D^2(F_A, F_B) = 2\,\mathbb{E}\left[\|A - B\|\right] - \mathbb{E}\left[\|A - A'\|\right] - \mathbb{E}\left[\|B - B'\|\right] \tag{3.14}$$

with $\mathbb{E}$ the expectation value. To compute the empirical energy distance let $\boldsymbol{S}_A$ be a sample set with m samples and $\boldsymbol{S}_B$ a sample set with size n. The empirical energy distance is defined as

$$Cor_{ED}(\boldsymbol{S}_A, \boldsymbol{S}_B) = \frac{2}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left\|a_i - b_j\right\| - \frac{1}{m^2} \sum_{i=1}^{m} \sum_{j=1}^{m} \left\|a_i - a_j\right\| - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left\|b_i - b_j\right\| . \tag{3.15}$$

A really useful property is that no pair-wise correlations between $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ are required to calculate the empirical energy distance. Even more, $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ do not even have to have the same number of events. This allows computing the correlation score without doing the actual correlation. Consequently, only event pairs with a low energy distance should be processed by a matcher lowering the computational cost significantly.

To obtain a p-value for the energy distance, a permutation test is done. As no pairs between $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ are assumed both samples sets have to be permuted. Therefore, $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$ are combined into a single sample set, randomly permuted and then split again into $\boldsymbol{S}_A^\pi$ and $\boldsymbol{S}_B^\pi$. It is important to note that $\boldsymbol{S}_A^\pi$ and $\boldsymbol{S}_B^\pi$ probably contain samples from $\boldsymbol{S}_A$ and $\boldsymbol{S}_B$. The test statistic is defined as

$$S(\boldsymbol{S}_A, \boldsymbol{S}_B) = T(\boldsymbol{S}_A, \boldsymbol{S}_A) + T(\boldsymbol{S}_A, \boldsymbol{S}_B) + T(\boldsymbol{S}_B, \boldsymbol{S}_A) + T(\boldsymbol{S}_B, \boldsymbol{S}_B) \tag{3.16}$$

with

$$T(\boldsymbol{S}_A, \boldsymbol{S}_B) = \frac{m \cdot n}{m + n} \mathrm{Cor}_{ED}(\boldsymbol{S}_A, \boldsymbol{S}_B) .$$

Based on this test statistic, the p-value is calculated as

$$p = \frac{1}{k} \sum_{i=1}^{k} \mathbb{1}\left(S(\boldsymbol{S}_A, \boldsymbol{S}_B) > S(\boldsymbol{S}_A^\pi, \boldsymbol{S}_B^\pi)\right) \tag{3.17}$$

with k the number of repetitions. This test statistics allows testing for independence using hypothesis tests. Furthermore, as no pairs between A and B are assumed, the test can be done before actually computing the correlated event pairs.

3.2 Iterative Closest Events

The first approach to compute the temporal lag between two event types is called Iterative Closest Events. This algorithm is an adaptation of the ICP algorithm [28]. ICP is a quite famous EM algorithm from computer vision for point set registration. Given two point clouds, ICP searches for a rigid transformation of one point cloud such that the distance between both point clouds is minimised. This method is often used in computer vision to reconstruct 3D scans of an object. Figure 3.2 visualises the point set registration task.

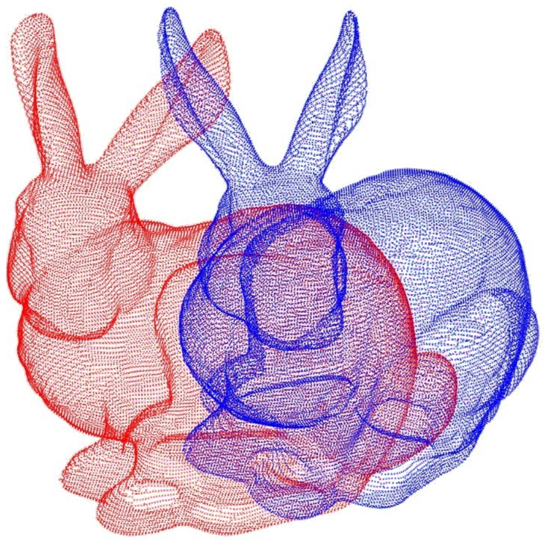


Figure 3.2: Example for point set registration. ICP computes a transformation of the blue point cloud such that the distance to the red point cloud becomes minimal [29].

Given two datasets called *data* and *reference*, the ICP algorithm searches for a transformation that minimises a given cost function. This optimal transformation is calculated iteratively using the following steps [28]:

1. For each point in *data* find the closest point in *reference*.
2. Estimate a rigid transformation that minimises the distance between all *data* - *reference* pairs.
3. Apply the estimated transformation to *data*.

Those steps are repeated until the rigid transformation is sufficiently small.

Let $data = \boldsymbol{S}_A$ be a sequence of events of type A with $|\boldsymbol{S}_A| = m$ the number of events of type A and $reference = \boldsymbol{S}_B$ a sequence of events of type B with $|\boldsymbol{S}_B| = n$ the number of events of type B.

***Data - Reference* Assignment**

For each point in $\boldsymbol{S}_A$ the corresponding point in $\boldsymbol{S}_B$ with minimal Euclidean distance has to be found. This calculation is normally done by computing all pairwise distances and selecting the smallest one. It is important to note that several points in $\boldsymbol{S}_A$ can be assigned to the same point in $\boldsymbol{S}_B$ and that some points in $\boldsymbol{S}_B$ can remain unpaired. Let b_{a_i} be the assigned point to a_i.

Transformation Estimation

With the pairwise correlations given, the optimal rigid transformation can be calculated. Therefore, the optimisation problem

$$\underset{r,t\,\in\,\mathbb{R}^n}{\text{minimise}} \quad \frac{1}{m}\left\|\sum_{i=0}^{m}(r\cdot a_i + t) - b_{a_i}\right\|_2 \tag{3.18}$$

with r being a rotation matrix and t being a translation vector has to be solved. For this optimisation problem an arbitrary solver can be used.

Originally, the ICP algorithm was developed to match three dimensional point clouds. However, for this application the problem can be simplified: It is possible to interpret *reference* and *data* as two one-dimensional vectors. Those vectors are not rotated against each other and only shifted in one direction. Consequently, the transformation estimation in Equation (3.18) can be simplified to

$$\underset{t\,\in\,\mathbb{R}}{\text{minimise}} \quad \frac{1}{m}\sum_{i=0}^{m}\left|(a_i + t) - b_{a_i}\right| \tag{3.19}$$

with t being a scalar. This optimisation problem can be efficiently solved using least squares. Additionally, the Jacobi matrix can be computed as

$$J(t,\boldsymbol{S}_A,\boldsymbol{S}_B) = \frac{1}{m}\sum_{i=0}^{m}\frac{(a_i + t) - b_{a_i}}{\left|(a_i + t) - b_{a_i}\right|} \tag{3.20}$$

and a constant Hessian matrix

$$H(t,\boldsymbol{S}_A,\boldsymbol{S}_B) = 0\,. \tag{3.21}$$

Naively implemented, these two steps run in $O(m\cdot n)$ as all pair-wise distances have to be calculated. However, several techniques exist to improve the *data-reference* assignment using $k-d$ trees or caching [30]. Furthermore, it can be proven that the number of iterations is also limited by $O(m\cdot n)$ as the input samples are one-dimensional [31].

ICE calculates one transformation for all *data* points. As a consequence outliers can influence the optimisation process. Depending on the type of outlier the influence differs:

Additional Point in $\boldsymbol{S}_A$ / Missing Point in $\boldsymbol{S}_B$

Assume a missing point in $\boldsymbol{S}_B$. If the missing point in $\boldsymbol{S}_B$ would not be matched by any point in $\boldsymbol{S}_A$, nothing happens. Otherwise, the cost function is overestimated as the distance to the missing point is smaller than the distance to the wrongly chosen one. The cost function will also be overestimated if an additional point is present in $\boldsymbol{S}_A$ as an additional distance term is introduced.

To counter the effects of additional points in $\boldsymbol{S}_A$, a random sampling of *data* is done before calculating the best pairs. If the rough number of outliers is known in advance, *data* can be simply trimmed. Let f be the expected number of outliers in percent. For each point in $\boldsymbol{S}_A$ find the closest in $\boldsymbol{S}_B$ and sort the pairs by their distance. Next, remove the f pairs with the highest distance [32]. However, if the amount of outliers is not known—as it is usually the case—this approach is not reasonable. For this situation it is better to adjust the used samples dynamically. A quite simple approach is performing plain outlier removal using z-scores and thresholding [33].

Additional Point in $\boldsymbol{S}_B$ / Missing Point in $\boldsymbol{S}_A$

This time assume an additional point in $\boldsymbol{S}_B$. If this point is not matched by any point in $\boldsymbol{S}_A$, nothing happens. Otherwise, the cost function is underestimated as the distance to the additional point is smaller than to the correct one. The cost function will also be underestimated if one point is missing in $\boldsymbol{S}_A$ as one distance term is missing. Unfortunately, additional points in $\boldsymbol{S}_{\boldsymbol{B}}$ cannot be easily sorted out.

As ICE is a special case of an EM algorithm, ICE suffers from local minima [34]. Consequently, it is important that optimisation starts close to the true solution to ensure a global minimum. Therefore, the *data* set is shifted by a "good" initial guess. To obtain a good initial guess, a *Random Sample Consensus* (RANSAC) [35] algorithm is used [36]:

1. Select n sample points randomly from $\boldsymbol{S}_A$. Ensure that all sample points have a minimal distance d to limit the effects of local distortions.
2. For each selected sample point find the k nearest neighbours in $\boldsymbol{S}_B$ and select one of these at random.
3. Based on these n pairs, compute the rigid transformation as usually in ICE.

These three steps are repeated q times and the transformation with the minimal error is used as initial guess. This method ensures that with very high probability the initial guess is not influenced by outliers. If a priori information about the true distribution is given, this knowledge can be used to enhance the initial guess. This allows basic incorporation of additional knowledge.

Finally, the optimal transformation and final assignments are computed. Based on the final assignments, the time lag distribution is calculated. For each event pair a sample point is generated. This set of samples is used to train a KDE with

$$P(B \mid A) = \hat{f}_h\left(\begin{bmatrix} b_{a_1} - a_1 + t & b_{a_2} - a_2 + t & \dots & b_{a_m} - a_m + t \end{bmatrix}\right) . \tag{3.22}$$

3.3 Linear Programming

The second approach is called *LpMatcher*. Like the *lagEM* algorithm, this algorithm calculates the assignment probability for all trigger-response pairs. Yet, instead of using an EM algorithm to solve this optimisation problem, the problem is formulated as a linear programming minimisation.

Let $\boldsymbol{S}_A$ be a sequence of events of type A with $|\boldsymbol{S}_A| = m$ being the number of events of type A and $\boldsymbol{S}_B$ a sequence of events of type B with $|\boldsymbol{S}_B| = n$ the number of events of type B. The problem is as follows:

$$\begin{aligned} \text{minimise} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in [0,1]^{m \cdot n} \end{aligned} \; . \tag{3.23}$$

x represents the final assignment and is initialised uniformly. The cost function $f(x)$ is designed to minimise the squared temporal distance

$$f(x) = x_{11}\,\frac{(a_1 - b_1)^2}{n-1} + x_{21}\,\frac{(a_2 - b_1)^2}{n-1} + \dots + x_{mn}\,\frac{(a_m - b_n)^2}{n-1} \; . \tag{3.24}$$

The function $g(x)$ is used to enforce the order of events. As a response cannot occur before the corresponding trigger, the time difference between two elements has to be negative, i.e.,

$$a_i - b_j \leq 0 \; . \tag{3.25}$$

Furthermore, these inequalities can be used to enforce that one event a_i can only create at most one event b_j

$$x_{i1} + x_{i2} + \dots + x_{in} \leq 1 \; . \tag{3.26}$$

The equality constraint is used to enforce that every response is created by exactly one trigger event

$$x_{1j} + x_{2j} + \dots x_{mj} = 1 \; . \tag{3.27}$$

This minimisation problem can be solved by any optimisation algorithm for linear cost functions. As soon as all assignments x_{ij} are calculated, a distribution for the time lag is estimated. Therefore all pairs with $x_{ij} = 1$ are used to train a KDE with

$$P(B \mid A) = \hat{f}_h\left(\begin{bmatrix}(b_j - a_i) \cdot x_{ij} & (b_l - a_k) \cdot x_{kl} & \dots & (b_n - a_m) \cdot x_{mn}\end{bmatrix}\right) \; . \tag{3.28}$$

In the next section, a mathematically sound formulation of the optimisation is given and all equations are derived.

3.3.1 Mathematically Sound Formulation

At first, the cost function is derived. Next, both constraints are added to form a linear minimisation problem. Finally, the minimisation problem is transformed to make it computational feasible.

Cost Function

The original goal of the matching process is to find pairs of events a_i and b_j such that the time lag of each pair is roughly identical. This is equivalent to minimising the variance.

Definition 9 (Variance)
Let X be a random variable with realisations x_i. The variance of X is defined as

$$\begin{aligned}\operatorname{Var}(X) &= \mathbb{E}\left((X - \mathbb{E}(X))^2\right) = \mathbb{E}\left(X^2\right) - \left(\mathbb{E}(X)\right)^2 \\ &= \frac{\sum\limits_{i=1}^{n} x_i^2}{n} - \left(\frac{\sum\limits_{i=1}^{n} x_i}{n}\right)^2 = \frac{\sum\limits_{i=1}^{n} x_i^2}{n} - \frac{(\sum\limits_{i=1}^{n} x_i)^2/n}{n} \; .\end{aligned} \tag{3.29}$$

During the event correlation only samples of the true distribution are known, i.e. the time differences. Consequently, it is not possible to calculate the variance over all possible values, but only the sampled ones. However, the variance as defined in Equation (3.29) is biased if calculated over a sample set [37]. Using Bessel's Correction [38] sample variance is defined as

$$\mathrm{Var}(X) = \frac{\sum\limits_{i=1}^{n} x_i^2}{n-1} - \frac{\left(\sum\limits_{i=1}^{n} x_i\right)^2 / n}{n-1} . \tag{3.30}$$

To compute the variance, pairwise distances of events have to be used. As it is not known which events are correlated, all pairs have to be considered. This can be formulated as a matrix

$$\Delta = \begin{bmatrix} a_1 - b_1 & a_2 - b_1 & \dots & a_m - b_1 \\ a_1 - b_2 & a_2 - b_2 & \dots & a_m - b_2 \\ \vdots & \vdots & \dots & \vdots \\ a_1 - b_n & a_2 - b_n & \dots & a_m - a_n \end{bmatrix} . \tag{3.31}$$

Accordingly, an assignment variable $x \in \{0,1\}^{m \cdot n}$ is introduced. This variable models whether a_i and b_j are considered as correlated or not.

Corollary 1. *Let $d_{ij} = a_i - b_j$. Using Δ, the variance can be formulated as a matrix multiplication.*

$$\begin{aligned} f(x) &= \frac{\left(vec(\Delta)^2\right)^T \cdot x}{n-1} - \frac{x \cdot F \cdot x^T}{n-1} \\ &= \frac{\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{n-1} - \frac{\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij} x_{ij}\right)^2 / n}{n-1} = \mathrm{Var}(\Delta, x) \end{aligned} \tag{3.32}$$

with

$$\begin{aligned} F &= \frac{\mathrm{vec}(\Delta) \cdot \mathrm{vec}(\Delta)^T}{n} \\ &= \frac{1}{n} \cdot \begin{bmatrix} d_{11} \cdot d_{11} & d_{11} \cdot d_{21} & \dots & d_{11} \cdot d_{mn} \\ d_{21} \cdot d_{11} & d_{21} \cdot d_{21} & \dots & d_{21} \cdot d_{mn} \\ \vdots & \vdots & \dots & \vdots \\ d_{m1} \cdot d_{11} & d_{m1} \cdot d_{21} & \dots & d_{mn} \cdot d_{mn} \end{bmatrix} . \end{aligned}$$

Proof. The proof is divided into two parts for the two different terms.

$$\frac{\left(\operatorname{vec}(\Delta)^2\right)^T \cdot x}{n-1} = d_{11}^2 \cdot x_{11} + d_{12}^2 \cdot x_{12} + \cdots + d_{mn}^2 \cdot x_{mn} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 \cdot x_{ij}}{n-1}$$

$$\begin{aligned}
\frac{x \cdot F \cdot x^T}{n-1} &= \frac{x_{11} \cdot (F_{11}x_{11} + \cdots + F_{m1}x_{m1}) + \cdots + x_{mn} \cdot (F_{m1}x_{m1} + \cdots + F_{mn}x_{mn})}{n-1} \\
&= \frac{x_{11} \cdot \left(\frac{d_{11}^2}{n}x_{11} + \cdots + \frac{d_{m1}d_{11}}{n}x_{m1}\right) + \cdots + x_{mn} \cdot \left(\frac{d_{m1}d_{mn}}{n}x_{m1} + \cdots + \frac{d_{mn}^2}{n}x_{mn}\right)}{n-1} \\
&= \frac{\frac{1}{n}\left(x_{11} \cdot \left(d_{11}^2 x_{11} + \cdots + d_{m1}d_{11}x_{m1}\right) + \cdots + x_{mn} \cdot \left(d_{m1}d_{mn}x_{m1} + \cdots + d_{mn}^2 x_{mn}\right)\right)}{n-1}
\end{aligned}$$

It is important to note that $x_{ij} \in \{0, 1\}$. This means that only some $d_{ij} \cdot d_{kl}$ terms are used, depending on x_{ij} and x_{kl}. Erase from the equation above all $d_{ij} \cdot d_{kl}$ where either $x_{ij} = 0$ or $x_{kl} = 0$. Furthermore, all remaining x_{ij} can be eliminated as they are all 1.

$$\begin{aligned}
\frac{x \cdot F \cdot x^T}{n-1} &= \frac{\frac{1}{n}\left(d_{ij}^2 + d_{ij}d_{kl} + d_{kl}d_{ij} + d_{kl}^2 + \ldots\right)}{n-1} \\
&= \frac{\frac{1}{n}\left(d_{ij}^2 + 2d_{ij}d_{kl} + d_{kl}^2 + \ldots\right)}{n-1} \\
&= \frac{\frac{1}{n}\left(d_{ij} + d_{kl} + \ldots\right)^2}{n-1} \\
&= \frac{\left(\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}x_{ij}\right)^2 / n}{n-1}
\end{aligned}$$

□

With the above definition of the cost function, the event-correlation problem can be solved by quadratic optimisation algorithms, as the sample variance is a quadratic cost function. However, solving quadratic functions is computationally not easy [39]. Therefore, an approximation is applied to linearise the cost function. The approximation is done by using only the squared distances

$$f(x) = \frac{\left(\operatorname{vec}(\Delta)^2\right)^T \cdot x}{n-1} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 x_{ij}}{n-1} . \tag{3.33}$$

As the quadratic term of the variance

$$\frac{\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}x_{ij}\right)^2 / n}{n-1} \geq 0 \tag{3.34}$$

$\forall d \in \mathbb{R}$, $x_{ij} \in \{0,1\}$, the approximation is an upper bound for the variance. To evaluate the influence of this approximation on the cost function, an approximation ratio α is calculated

$$\alpha = \frac{\frac{\left(\mathrm{vec}(\Delta)^2\right)^T \cdot x}{n-1}}{\frac{-x \cdot F \cdot x^T}{n-1}} = \frac{\left(\mathrm{vec}(\Delta)^2\right)^T \cdot x}{-x \cdot F \cdot x^T} = \frac{\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}x_{ij}\right)^2 / n} . \tag{3.35}$$

As α depends on d_{ij} and x_{ij}, α depends on a concrete problem. So instead of calculating a specific α, the bound of α is calculated. Using the *Cauchy-Schwarz Inequality* [40], an upper bound for α can be deduced.

Theorem 1 (Cauchy-Schwarz Inequality)
Let $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ be two sequences of real numbers, then

$$\left(\sum_{i=1}^{n} a_i b_i\right)^2 \leq \sum_{i=1}^{n} a_i^2 \cdot \sum_{i=1}^{n} b_i^2 . \tag{3.36}$$

Applying this Theorem to α, an upper border is obtained.

$$\begin{aligned}
\alpha &= \frac{\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}x_{ij}\right)^2 / n} \\
&= \frac{n \cdot \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} 1 \cdot d_{ij}x_{ij}\right)^2} \\
&\leq \frac{n \cdot \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-\left(\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} 1^2 \cdot \sum\limits_{i=1}^{m}\sum\limits_{j=1}^{n} (d_{ij}x_{ij})^2\right)}
\end{aligned} \tag{3.37}$$

$$
\begin{aligned}
&= \frac{n \cdot \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-m \cdot n \cdot \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}^2} \\
&\leq \frac{\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-m \cdot \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}} \\
&= \frac{1}{-m}
\end{aligned}
$$

The second inequality holds for all $x_{ij} \in [0, 1]$. More specifically, even equality is given as $x_{ij} \in \{0, 1\}$ by definition. Using this approximation for α, the relation between the approximate cost function and original cost function can be calculated.

$$
\begin{aligned}
& & \alpha &\leq \frac{1}{-m} \\
&\iff & \frac{\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{-\left(\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij} x_{ij}\right)^2 / n} &\leq \frac{1}{-m} \\
&\iff & \frac{-\left(\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij} x_{ij}\right)^2}{n \cdot \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}} &\geq -m \\
&\iff & \frac{-\left(\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij} x_{ij}\right)^2}{n} &\geq -m \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 x_{ij} \\
&\iff & \frac{\left(\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij} x_{ij}\right)^2}{m \cdot n} &\leq \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}^2 x_{ij}
\end{aligned}
\tag{3.38}
$$

The left side of the inequality represents the quadratic term that was eliminated from the original cost function and the right term the approximated cost function. It is important to note that the quadratic term is smaller than the linear term and converges to 0 with $m,\ n \to \infty$. For large m and n the quadratic term will influence the solution only minimal and the cost function is dominated by the linear term. This effect is visualised in Figure 3.3.

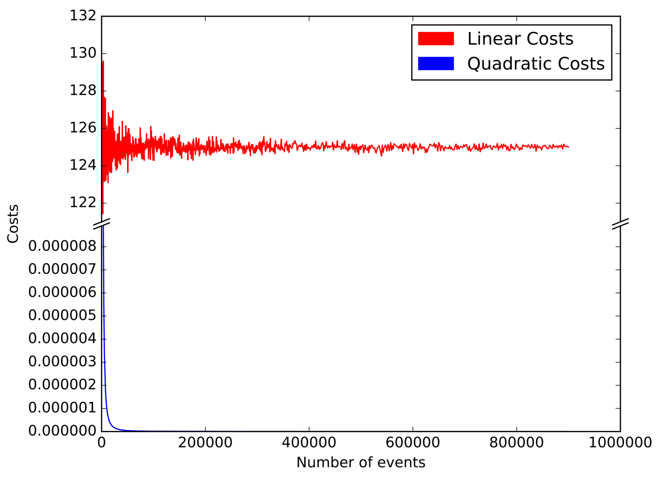


Figure 3.3: Visualisation of the estimation error for the linear and adjusted quadratic term of the variance. As correlation data set, n samples from a normal distribution $\mathcal{N}(10, 5)$ are used. It is obvious that with increasing number of events the quadratic cost term becomes irrelevant.

Integer Linear Programming

This section formulates the event correlation problem as an *Integer Linear Programming* (ILP) problem. Therefore, first some basic definitions are provided.

Definition 10 (Linear Function)
Let $f(x) : \mathbb{R}^n \to \mathbb{R}$ be a function. f is called linear if it fulfils the following properties:

- *Additivity: $f(x + y) = f(x) + f(y)$*
- *Homogeneity of degree 1: $f(\alpha \cdot x) = \alpha \cdot f(x)$*

Based on linear functions, Integer Linear Programming can be defined.

Definition 11 (Integer Linear Programming)
Let $f(x) : \mathbb{R}^n \to \mathbb{R}$ be a linear function over the variable x. Let $g(x) \leq 0$ be an inequality constraint and $h(x) = 0$ be an equality constraint. An integer linear minimisation problem is thus written as

$$\begin{aligned} \textit{minimise} \quad & f(x) \\ \textit{subject to} \quad & g(x) \leq 0 \\ & h(x) = 0 \\ & x \in \mathbb{Z}^n \,. \end{aligned} \tag{3.39}$$

$f(x)$ can be rewritten such that $f(x) = c^T \cdot x$ is represented as a vector product.

The constraint $x \in \mathbb{Z}^n$ extends *Linear Programming* (LP) to ILP. In the context this work, x can be even more limited to $x \in \{0,1\}^n$. Usually, linear optimisation is formulated without an equality constraint. Yet, the extension to equality constraints is simple as an equality constraint can be rewritten as two inequalities

$$a \cdot x = b \Leftrightarrow a \cdot x \leq b \wedge -(a \cdot x) \leq -b\,. \tag{3.40}$$

According to Definition 11 a linear cost function, an inequality constraint and an optional equality constraint have to be provided for ILP. In the previous section, the cost function $f(x)$ was already derived as Equation (3.33). It only remains to prove that $f(x)$ is linear.

Corollary 2. *The cost function*

$$f(x) = \frac{\sum\limits_{i=1}^{m} \sum\limits_{j=1}^{n} d_{ij}^2 x_{ij}}{n-1}$$

is a linear function.

Proof. As a linear function has to be additive and have homogeneity of degree 1, this proof is split into two parts. For simplicity substitute $c_{ij} = \frac{d_{ij}^2}{n-1}$.

Proof for additivity.

$$\begin{aligned} f(x+y) =& (x_{11}+y_{11}) \cdot c_{11} + (x_{21}+y_{21}) \cdot c_{22} + \cdots + (x_{mn}+y_{mn}) \cdot c_{mn} \\ =& x_{11} \cdot c_{11} + y_{11} \cdot c_{11} + x_{21} \cdot c_{21} + y_{21} \cdot c_{22} + \cdots + x_{mn} \cdot c_{mn} + y_{mn} \cdot c_{mn} \\ =& (x_{11} \cdot c_{11} + x_{21} \cdot c_{21} + \cdots + x_{mn} \cdot c_{mn}) + (y_{11} \cdot c_{11} + y_{21} \cdot c_{21} + \cdots + y_{mn} \cdot c_{mn}) \\ =& f(x) + f(y) \end{aligned}$$

Proof for Homogeneity of degree 1.

$$\begin{aligned} f(\alpha x) &= (\alpha \cdot x_{11}) \cdot c_{11} + (\alpha \cdot x_{21}) \cdot c_{21} + \cdots + (\alpha \cdot x_{mn}) \cdot c_{mn} \\ &= \alpha \cdot (x_{11} \cdot c_{11} + x_{21} \cdot c_{21} + \cdots + x_{mn} \cdot c_{mn}) \\ &= \alpha \cdot f(x) \end{aligned}$$

□

Next, the two constraint functions $g(x)$ and $h(x)$ are defined.

Definition 12 (Inequality Constraints)
For better visibility the constraint $g(x)$ is divided into $g_1(x)$ and $g_2(x)$. Both parts can simply be stacked to create the actual $g(x)$. Let

$$\begin{aligned} g_1(x) &= \begin{bmatrix} d_{11}\, x_{11} \\ d_{21}\, x_{21} \\ \vdots \\ d_{mn}\, x_{mn} \end{bmatrix} \\ &= \begin{bmatrix} d_{11} & 0 & \dots & 0 \\ 0 & d_{21} & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & d_{mn} \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{mn} \end{bmatrix} = G_1 \cdot x \end{aligned} \tag{3.41}$$

with G_1 the matrix representation of $g_1(x)$. Furthermore, let

$$\begin{aligned} g_2(x) &= \begin{bmatrix} x_{11} + x_{12} + \dots + x_{1n} \\ x_{21} + x_{22} + \dots + x_{2n} \\ \vdots \\ x_{m1} + x_{m2} + \dots + x_{mn} \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 & 0 & 1 & \dots & 0 & \dots & 0 & 1 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots & \vdots & 0 & \ddots & \vdots & \dots & \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & \dots & 0 & 0 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{mn} \end{bmatrix} = G_2 \cdot x \end{aligned} \tag{3.42}$$

with G_2 the matrix representation of $g_2(x)$. To obtain the final $g(x)$, G_1 and G_2 have to be stacked.

The inequality G_1 enforces that the trigger has to occur before the response. G_2 is used to prevent one b_j being caused by several a_i's.

Definition 13 (Equality Constraint)
Let the equality constraint be defined as

$$
\begin{aligned}
h(x) &= \begin{bmatrix} \mathbb{1}_1(1)\cdot x_{11} + \mathbb{1}_1(2)\cdot x_{21} + \dots + \mathbb{1}_1(m)\cdot x_{m1} - 1 \\ \mathbb{1}_2(1)\cdot x_{12} + \mathbb{1}_2(2)\cdot x_{22} + \dots + \mathbb{1}_2(m)\cdot x_{m2} - 1 \\ \vdots \\ \mathbb{1}_n(1)\cdot x_{1n} + \mathbb{1}_n(2)\cdot x_{2n} + \dots + \mathbb{1}_n(m)\cdot x_{mn} - 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & \dots & 1 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 1 & 0 & \dots & 0 \\ \vdots & \dots & \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 & 1 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{mn} \end{bmatrix} \cdot \begin{bmatrix} \mathbb{1}_1(1) & \mathbb{1}_1(2) & \dots & \mathbb{1}_n(m) \end{bmatrix} - 1 \\
&= H \cdot x \cdot \mathbb{1} - 1
\end{aligned}
\tag{3.43}
$$

with H the matrix representation of h(x).

The indicator variable $\mathbb{1}_j(i)$ is used to enforce that event b_j can only be triggered by one a_i. $\mathbb{1}_j(i)$ can be one for at most one $i \in \{1, 2, \dots, m\}$. Basically, this enforces that only integer solutions are allowed with $x_{ij} \in \{0, 1\}$ meaning that one event either triggered another event or not. Consequently, the event correlation problem formulated as above can be solved by ILP algorithms.

LP-Relaxation

In the previous section it was shown that the event correlation problem can be formulated as an ILP problem. Yet, ILP is a *Non-Deterministic, Polynomial Time* (NP)-complete problem and therefore not efficiently solvable [41].

Corollary 3. *ILP is an NP-complete problem.*

Proof. To prove NP-completeness it has to be proven that ILP ∈ NP and ILP ∈ NP-hard.

1. ILP ∈ NP as the value for each x_{ij} can simply be guessed from $\{0, 1\}$ and all constraints checked in polynomial time yielding a valid solution.
2. To prove that ILP is NP-hard, SAT [1] is reduced to ILP. Let $z_{11}, \dots, z_{mn}$ be the variables of SAT with corresponding variables $x_{11}, \dots, x_{mn}$ in ILP and $x_{ij} \in \{0, 1\}$. Let $x_{ij} = 1$ if and only if

[1]The *Boolean Satisfiability Problem* (SAT) is a well known NP-hard problem. Let a Boolean formula in conjunctive normal form be given. SAT answers the question if an assignment of variables to the Boolean values *true* and *false* exists such that the formula evaluates to *true* [42].

$z_{ij} = true$ and $x_{ij} = 0$ if and only if $z_{ij} = false$. Furthermore,

$$\overline{z_{ij}} = \begin{cases} x_{ij} = 0 & \text{if } z_{ij} = true \\ x_{ij} = 1 & \text{if } z_{ij} = false \end{cases}$$

and $z_i \wedge z_j$ implies $x_i + x_j$. Replace each clause of SAT according to the above rules. SAT is satisfiable if and only if the sum of each clause is greater than zero. Consequently, a polynomial time reduction exists.

□

Usually to solve ILP, the integer constraint is ignored. Instead of requiring binary assignments $x_{ij} \in \{0, 1\}$, an assignment probability $x_{ij} \in [0, 1]$ is calculated. The relaxation of ILP with $x \in \mathbb{Z}^n$ to LP with $x \in \mathbb{R}^n$ is called LP-relaxation. In contrast to ILP, LP is solvable in polynomial time by many algorithms, for example the *Simplex* algorithm [43]. Even though the Simplex algorithm has a worst-case complexity of $O(2^n)$, it can be proven that the average runtime is polynomial [44]. Consequently, the LP-relaxation reduces the average runtime significantly.

Obviously, the LP-relaxation modifies and approximates the original problem. This approximation introduces an additional error called the *integrality gap* [45]. It has to be shown that the LP-relaxation does not increase the error too much. Let $\mathbb{EC}$ be the problem event correlation and $x^* = \text{ILP}(\mathbb{EC})$ the optimal ILP solution for $\mathbb{EC}$. Furthermore, let $z^* = \text{LP}(\mathbb{EC})$ be the optimal solution after the LP-relaxation.

Corollary 4. *The cost of the LP solution is a lower boundary for the cost of the ILP solution.*

$$f(z^*) \leq f(x^*) \tag{3.44}$$

Proof. Let $X = \{0, 1\}^n$ be the feasible space of the ILP problem and $Z = [0, 1]^n$ for the LP problem. The solution space of ILP is a subset of LP

$$X \subset Z\,.$$

□

As Corollary 4 shows, LP finds a lower border for the ILP problem as the feasible space of ILP is completely contained in the LP solution space. Figure 3.4 visualises this relationship.

An important observation is that LP is theoretically able to find integer solutions and therefore solving the original ILP problem. More specifically, LP always finds an integer solution if and only if the constraint matrices G and H are totally unimodular [46].

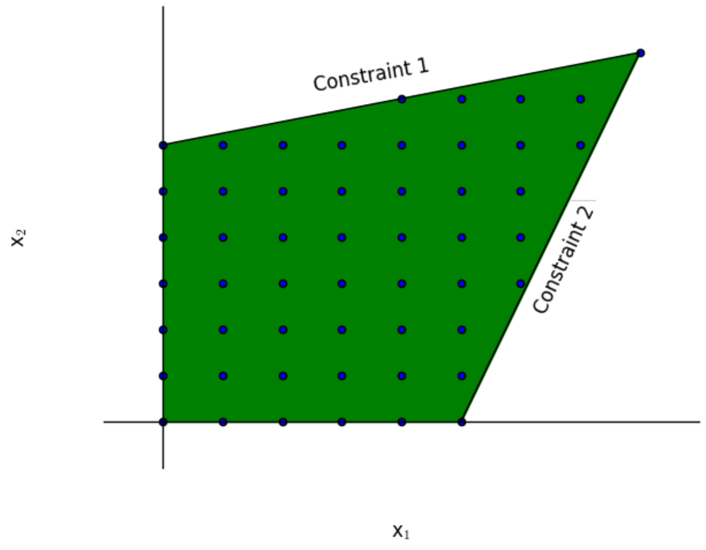


Figure 3.4: Feasible space for ILP and LP with two constraints. The green area shows the feasible space for LP and the blue dots the feasible solutions for ILP.

Definition 14 (Totally Unimodular)
A matrix A is called totally unimodular if every square sub-matrix of A is unimodular. A square matrix A is called unimodular if

$$\det(A) = \pm 1 \ . \tag{3.45}$$

So instead of estimating the approximation error, it is proven that the constraints are totally unimodular and consequently no approximation error exists. Therefore, the following corollary is helpful.

Corollary 5. *Any matrix A containing only $-1, 0$ and 1 is totally unimodular if each column of A contains at most two non-zero entries. Furthermore, all rows of A can be split into two sets such that two non-zero entries with the same sign from one column belong to different subset and two non-zero entries with different signs from one column belong to the same subset.*

See Theorem 7.3 in [46] for a proof of Corollary 5. Based on this corollary, G_2 and H are totally unimodular as they contain only 0 and 1 with only one 1 per column. However, G_1 has to be rewritten to fulfil the first condition.

$$G_1 = \begin{bmatrix} \mathrm{sgn}(d_{11}) & 0 & \dots & 0 \\ 0 & \mathrm{sgn}(d_{21}) & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \dots & 0 & \mathrm{sgn}(d_{mn}) \end{bmatrix} \tag{3.46}$$

with $\mathrm{sgn}(x)$ the signum function. In this way G contains at most two non-zero entries per column. Consequently, the LP algorithm always calculates an integer solution and the LP-relaxation does not introduce an additional error.

With all these approximations done, the event correlation can be solved by an arbitrary linear programming solver. It only remains to chose the initial guess for x. In the most basic case the assignments are uniformly initialised as

$$x_{ij} = 1/n \, . \tag{3.47}$$

However, it is easily possible to model any additional information into the initial guess. For example the event life time can be modelled by an exponential distribution or a priori knowledge about some event pairs can be included. Furthermore, additional knowledge can be modelled by introducing new constraints, e.g. a maximum event life time with $|d_{ij}| \leq \Theta$. This makes the *LpMatcher* an powerful, easily extended matcher for event correlation.

3.4 Dependency Networks

The previously described methods ICE and *LpMatcher* can be used to calculate the temporal lag distribution between two event types. However, in reality systems do not consist of pair-wise dependencies. Instead, one event type can depend on multiple other event types while being trigger for other events. An obvious modelling of these dependencies is a graph. This section describes how to convert pair-wise correlations to a meaningful network using graph and probability theory.

3.4.1 Correlation Graphs

The easiest way to model dependencies between several events is a directed graph. Let a sequence $\boldsymbol{S}$ with a set of event types $E_1, \dots, E_n$ be given. Each event type can be interpreted as a node in a graph. Two nodes are connected by an edge if their events are considered correlated by the energy distance. Furthermore, the edge can be annotated with additional information like the temporal lag computed by the *LpMatcher* or ICE. The resulting graph is a compact representation of all correlations in a system. However, for manual inspection these graphs may still be too complicated. Often it is desirable to change the abstraction level by filtering out less meaningful correlations.

A correlation is considered to be meaningful if knowing the time stamp of the trigger event provides precise knowledge about the time stamp of the response event. Correspondingly, a bad correlation provides imprecise information about the response time stamp given the trigger time. This goodness can be modelled by mutual information as described in Section 3.1. With this score for each edge, a method for removing edges can be defined. Therefore, all mutual information is collected and their 25% percentile $Q_{0.25}$ and 75% percentile $Q_{0.75}$ are derived. Next, a dynamic threshold Θ is defined as

$$\Theta = 2 \cdot (Q_{0.25} - (0.5 + \epsilon) \cdot (Q_{0.75} - Q_{0.25})) \tag{3.48}$$

with $\epsilon \in [0, 1]$ a given sensitivity parameter. Finally, remove all edges with $\mathrm{Cor}_{MI}(\boldsymbol{S}_A, \boldsymbol{S}_B) < \Theta$. The parameter ϵ is given by the user and defines the amount of edges to be removed. This allows the user to specify the trade-off between simplicity and precision of the dependency graph as desired. Even though these graphs are helpful for a manual analysis, they do not allow precise reasoning about the correlation of multiple event types.

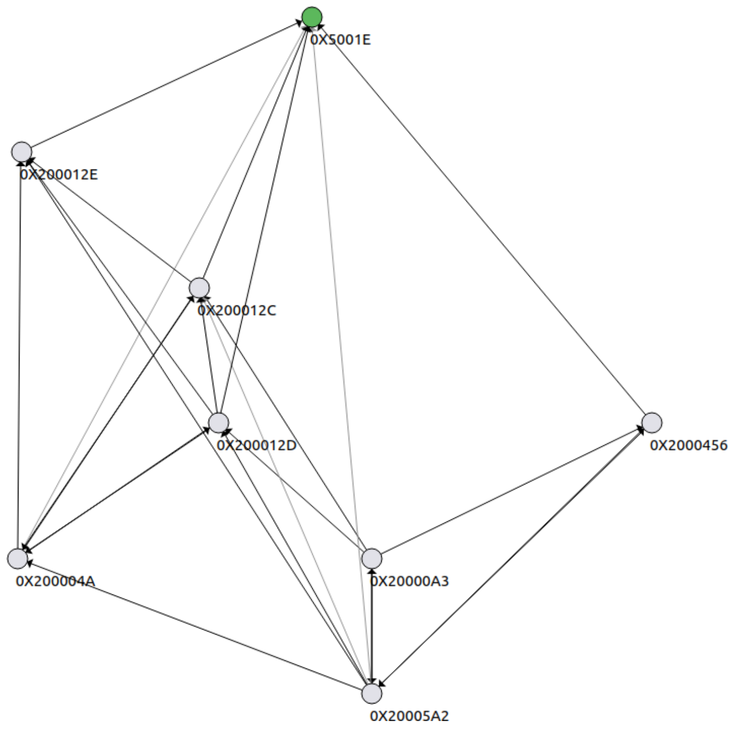


Figure 3.5: A correlation graph of eight event types. The edges show a correlation between two event types. Edges in grey were removed from the graph as their score is too low.

3.4.2 Bayesian Networks

Bayesian networks are a probabilistic graphical model to obtain knowledge about a random variable conditioned on other random variables. By utilising independence of random variables, a compact representation of dependencies can be achieved. This allows computation of probabilities given a set of observations without having to evaluate an exponential set of variables [47].

Definition 15 (Bayesian Network)
A Bayesian network is a tuple $N = (G, \Theta)$ with

- *G called the structure of N. The structure is a Directed Acyclic Graph* (DAG) *with nodes representing variables.*
- *Θ called the parametrization of N. The parameters are a conditional probability table for each variable. This table contains the probabilities of the respective variable conditioned on all parents.*

In the context of this work, Bayesian networks can be used to calculate the occurrence probability of an event with a specific type. In contrast to methods explained so far, Bayesian networks can model events triggered by multiple other events. This section shows that it is possible to create a Bayesian network with only the information given from the previous steps.

Every event type represents exactly one variable. This variable can be either true if the corresponding event occurred or false if not. These probabilities are know as $P(E_i)\ \forall E_i \in E$. For every variable three sets are defined:

- Parents(E_i) is the set of all variables with a direct edge leading to E_i.
- Descendants(E_i) is the set of all variables that can be reached from E_i. This may also happen indirectly.
- Non-descendants(E_i) are all variables not included in Parents(E_i) and Descendants(E_i).

Using ICE, *LpMatcher* or *lagEM*, the conditional probability $P(E_j \mid E_i)\ \forall E_i, E_j \in E$ between two variables is calculated as shown in Equations (3.22) and (3.28). Using these two sets of probabilities, a Bayesian network can be build in two steps [47].

Structure Learning
The first step is to learn the graph G of the Bayesian network. In general, each node represents one event type. Next, meaningful edges are added to create a DAG. For this task several approaches exist. A quite famous approach is the *Grow-Shrink strategy* [48]:

1. For each variable $E_i \in E$ compute a Markov blanket $\boldsymbol{B}(E_i)$ using only pair-wise tests for independence.
2. Compute graph structure using $\boldsymbol{B}(E_i)\ \forall E_i \in E$.
3. Remove edges from the graph forming a circle.

For a detailed description see [48]. The advantage of this algorithm is that only knowledge about pairwise independence of random variables is necessary. Originally, this is done by calculating Pearson χ^2 test of independence [49]. However, this can also be done by hypothesis testing using the energy distance.

During the previous steps of the event correlation framework, a dependency graph was already computed as shown in Section 3.4.1. The only problem is that this graph may contain cycles. Consequently, it is easier to use the already calculated structure and remove all edges forming a circle from it. This problem is equivalent to finding the feedback arc set, a well known problem from graph theory. For this task several exact and approximate algorithms exist, e.g. [50].

Parameter learning
With the structure of the Bayesian network given, the probability annotations have to be calculated. However, this is not possible yet as one event can depend on multiple parents—namely $P(X \mid E_i, E_j)$ with $X, E_i, E_j \in E$. Yet, it is possible to break down the equation into known parts:

$$\begin{aligned} P(X \mid E_i, E_j) &= \frac{P(E_i, E_j \mid X) \cdot P(X)}{P(E_i, E_j)} \\ &= \frac{P(E_i \mid X) \cdot P(E_j \mid X) \cdot P(X)}{P(E_i, E_j)} \\ &= \frac{P(X \mid E_i) \cdot P(E_i) \cdot P(X \mid E_j) \cdot P(E_j)}{P(X) \cdot P(E_i, E_j)} \\ &= \frac{P(X \mid E_i) \cdot P(E_i) \cdot P(X \mid E_j)}{P(X) \cdot P(E_i \mid E_j)} \end{aligned} \tag{3.49}$$

It is important to note that this equation only holds if E_i and E_j are conditionally independent given X. Even though this may not hold, conditional independence of E_i and E_j given X is assumed. Using this assumption, every term in Equation (3.49) except $P(E_i \mid E_j)$ is known for sure. If no correlation between E_i and E_j was detected during the previous analysis, $P(E_i \mid E_j)$ is unknown. In this case a uniform distribution is assumed as knowing E_j does not provide information about E_i.

Finally, using the *chain rule for probabilities* it is possible to recursively replace all probabilities of the kind $P(X \mid E_1, E_2, \ldots, E_n)$ as

$$P(X \mid E_1, E_2, \ldots, E_n) = \frac{P(X \mid E_1) \cdot P(E_1) \cdot P(X \mid E_2) \cdot P(E_2) \cdot \ldots \cdot P(X \mid E_n) \cdot P(E_n) \cdot P(X)}{P(X)^n \cdot P(E_1 \mid E_2, \ldots, E_n) \cdot P(E_2 \mid E_3, \ldots, E_n) \cdot \ldots \cdot P(E_{n-1} \mid E_n) \cdot P(E_n)} . \tag{3.50}$$

Based on the learned structure and parameters it is possible to actually calculate the occurrence probability of an event using inference. Inference is the process of reasoning about an unknown random variable based on observations of other random variables—called the *evidence* $\mathcal{E}$. Even though exact inference is NP-hard [51], an exact inference algorithm—namely *variable-elimination* [52]—is used for simplicity. This algorithm calculates $P(X \mid \mathcal{E})$ by enumerating over all possible values of the relevant random variables not included in $\mathcal{E}$ and caching intermediate results.

As a simplification, the actual occurrence time stamp of all event types is not modelled in the Bayesian network. Instead all events are reduced to random variables stating whether this event has occurred or not. The conditional probability $P(E_2 \mid E_1)$ is computed from the training sequence as

$$P(E_2 \mid E_1) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}\left(\mathcal{T}_\Delta\left(E_{2_i} - E_{1_j}\right) \geq \theta\right)}{\left|\boldsymbol{S}_{E_1}\right|} \tag{3.51}$$

with θ a minimum probability threshold. This simplification implies that all events in $\mathcal{E}$ and X occur roughly at the same time. In a future work, this implication should be replaced by equations actually enforcing the temporal coherence implied by the temporal lag.

Chapter 4

Experiments & Results

In the previous chapter two new algorithms called ICE and *LpMatcher* were introduced in the context of a generic event correlation framework. This chapter does a performance evaluation for synthetic and real word data. Furthermore, the performance of the *lagEM* algorithm is measured for the same tasks. The correlated event types are identified using the energy distance described in Section 3.1. If possible, all detected correlations are evaluated against a ground truth. This is done visually and using the energy distance.

4.1 lagEM Algorithm

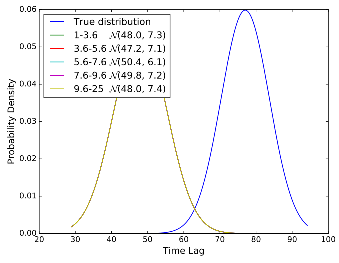
To get a reasonable comparison of the performance of the *lagEM* algorithm, a synthetic test scenario described by Zeng et al. (see [1] Table I) was reproduced. For the first test run, the following parameters for sequence generation were used:

> A sequence contains 200 elements in total. Event A occurs based on an exponential distribution with $\beta = 1/\lambda = 27.5$. Event B follows after event A with a normally distributed time lag with $\mu = 77.01$ and $\sigma = \sqrt{44.41} = 6.664$.

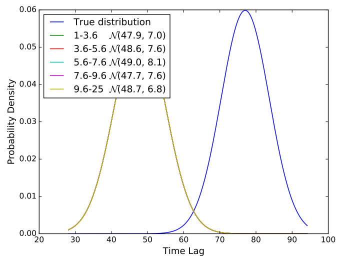
The first trials revealed that the performance of the *lagEM* algorithm highly depends on its initialisation. This is a general problem of EM algorithms [53]. Zeng et al. described the initialisation as "initialise μ and σ randomly" [1], without providing details of the exact procedure. It is mentioned that the algorithm suffers from local minima leading to wrong results. To compensate this problem, the *lagEM* algorithm is executed 200 times and results are filtered by their likelihood and averaged.

The obvious solution would be to choose μ and σ from an interval that is *sufficiently* large and hope that by chance some initialisations are close to the true solution. However, as the interval has to be large to contain many distributions, all probabilities are very small. This leads to numerical underflows and therefore to divisions by zero.

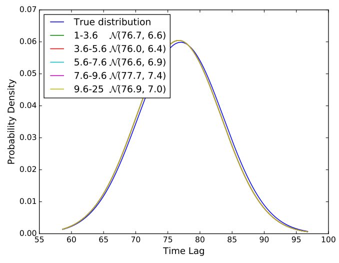
To evaluate the effect of an incorrect initialisation of the parameters on the result, a small experiment series was done. Therefore, the initial μ and σ were drawn from several uniform distributions. The actual boundaries are available in Appendix A. Based on these different configurations, the estimates of μ and σ are calculated. The results are visible in Figure 4.1. For each μ interval all σ are aggregated in one plot. It is remarkable, that all estimated distributions for one μ interval are nearly identical, independent of the initialisation of σ. Furthermore, the final estimate for μ highly depends on the initialisation as it is also visible in Figure 4.1. Only if $\mu \sim \mathcal{U}(62, 92)$ the *lagEM* algorithm is able to calculate the correct parameters.



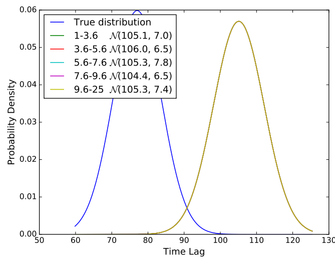
(a) $\mu \sim \mathcal{U}(0, 42)$.



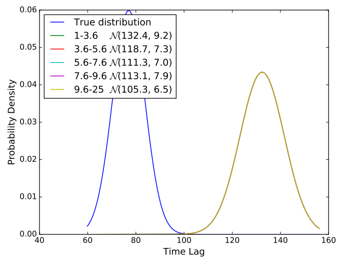
(b) $\mu \sim \mathcal{U}(42, 62)$.



(c) $\mu \sim \mathcal{U}(62, 92)$.



(d) $\mu \sim \mathcal{U}(92, 112)$.

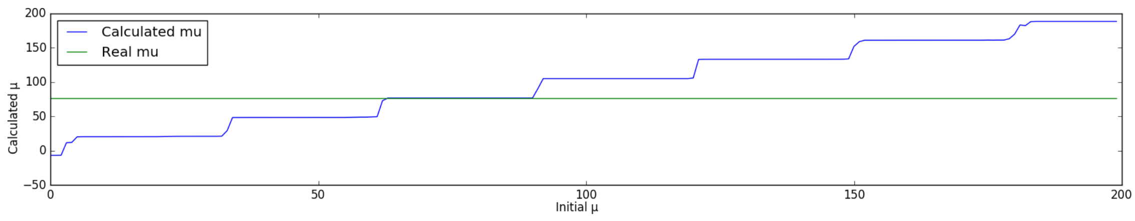


(e) $\mu \sim \mathcal{U}(112, 200)$.

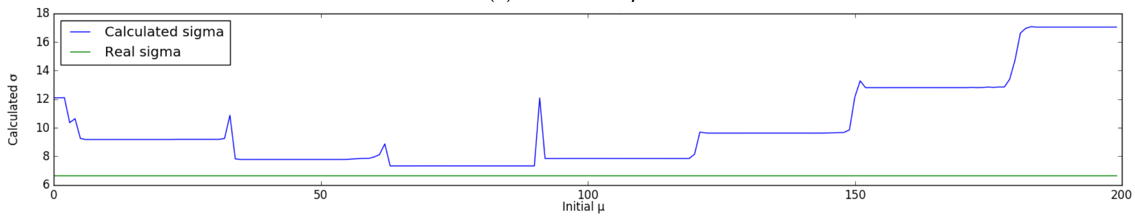
Figure 4.1: Calculated distributions for different initialisations of μ and σ. All σ intervals are aggregated for every μ interval. The five calculated distributions are all nearly identical. Final parameters are displayed in braces in the legend.

To further evaluate the effect of the initialisation, a second experiment was done. During this experiment σ was chosen from an uniform distribution $\sigma \sim \mathcal{U}(3, 28)$ and $\mu = 0, 1 \ldots, 199$. As in the first experiment, μ and σ were estimated. As it is visible in Figure 4.2, an initialisation of $\mu \in [62, 91]$ converges to the correct solution. Due to the various repetitions of the *lagEM* algorithm, this interval can be even further extended.

The performance for longer sequences could not be assessed as the runtime was too long. For a sequence with 1000 events and a good initialisation, the calculation required more than 2 hours. Furthermore, the approximation *appLagEM* also described in [1] did not speed up the calculation significantly.



(a) Estimated μ.



(b) Estimated σ.

Figure 4.2: Calculated μ and σ for μ initialised as $\mu = 0, 1, \ldots, 199$ and $\sigma \in \mathcal{U}[3, 28]$.

4.2 Synthetic Sequences

With a basic understanding of the *lagEM* algorithm given, all algorithms are evaluated for different scenarios. Yet, at first a tool for synthetic sequence generation with correlated events was implemented. The tool has to be able to create sequences with variable length and event types. Events should be created based on arbitrary distributions with the possibility that events are not recorded.

After implementing the synthetic code generation tool, a visualisation was added to understand longer, complex sequences. The visualisation displays events as circles including one letter to show the event type. '_' is used if no event occurred and '* ' if the event was not recorded. Arrows are included to show how events are correlated. These arrows are coloured on a grey-scale by the probability, that two events are correlated.

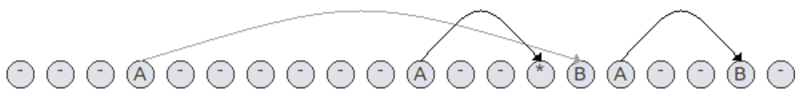


Figure 4.3: Screenshot of sequence visualiser.

With this tool given, several synthetic sequences were analysed to compare the performance of the three algorithms. As described in Section 4.1, the mean of the *lagEM* algorithm has to be initialised close to the true solution. For all following experiments, the initial mean is generated from a uniform distribution $\mathcal{U}(0, 100)$ and all correct means are also in [0, 100]. Even though this makes the task easier, the results of the *lagEM* algorithm would otherwise not be reasonable.

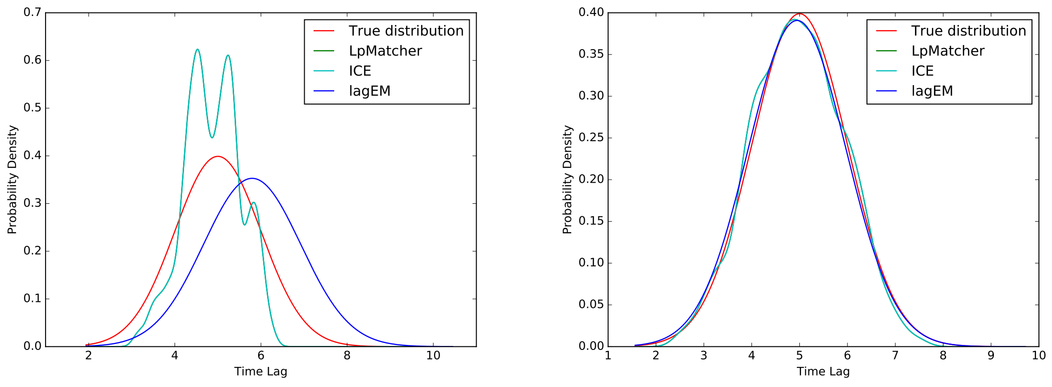
In the following four different scenarios are presented. Each scenario is more difficult than the previous ones. Furthermore, all scenarios were repeated with three randomly generated sequences and the results averaged.

Scenario 1

At first, a very simple scenario was tested:

> A sequence contains 100 elements in total. Event A occurs based on an uniform distribution with $\mathcal{U}(20, 30)$. Event B follows after event A with a normally distributed time lag $\mathcal{N}(5, 1)$. Furthermore, no events are lost.

In this scenario, trigger and response event occur in alternating order. An overlap between the event pairs does not happen and for each element an according event of the other type exists. As expected all algorithms are able to identify the correct event pairs. Figure 4.4a shows the estimated distributions for all algorithms. The estimates for ICE and the *LpMatcher* are identical both yielding an energy distance of 0.0364. As only 50 event pairs are included in the sequence, the estimated distribution is a bit noisy and does not fit the true distribution very well.



(a) Results for scenario 1 with 100 events. (b) Results for scenario 1 with 500 events.

Figure 4.4: Estimated distributions for scenario 1.

To compensate this distortion, the scenario is repeated with 500 events. As Figure 4.4b shows, the kernel density estimate looks much smoother resulting in smaller energy distances. Also the *lagEM* algorithm computes a better estimate improving the distance from 0.0618 to 0.0012. To resemble the true distribution even more, the sequence length has to be further increased.

Scenario 2

In the second scenario, an experiment from [1] was reused:

> A sequence contains 200 elements in total. Event A occurs based on an exponential distribution with $\beta = 1/\lambda = 27.5$. Event B follows after event A with a normally distributed time lag with $\mu = 77$ and $\sigma = \sqrt{44.41} = 6.664$. Again, no events are lost.

Figure 4.5 shows the estimated distributions for all three matchers. Even though nearly all event pairs overlap, ICE (0.0337) and *LpMatcher* (0.0063) are still able to estimate the true distribution. However, ICE does not produce consistent results requiring a few runs to obtain a good estimate. Even with several retries, *lagEM* was not able to estimate the correct distribution. This is also visible in the calculated distance of 0.4735. Zeng et al. reported the same behaviour but were able to solve it by using longer sequences [1].

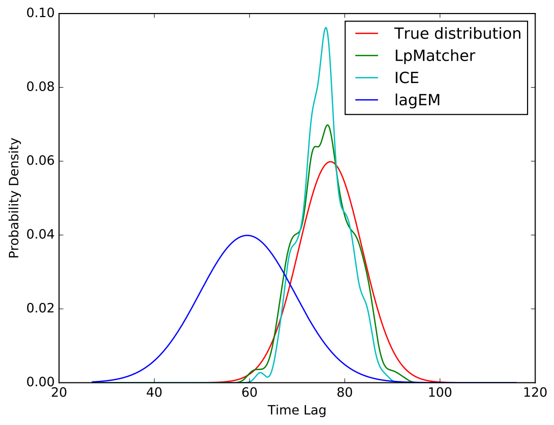


Figure 4.5: Estimated distributions for scenario 2.

Scenario 3

This scenario is an adaptation of scenario 2. In contrast to the previous scenarios, an event can be lost:

> A sequence contains 200 elements in total. Event A occurs based on an exponential distribution with $\beta = 1/\lambda = 27.5$. Event B follows after event A with a normally distributed time lag $\mathcal{N}(77.01, 6.664)$. In contrast to Scenario 2, both event types are lost with a probability of 10%.

As this scenario is an adaptation of scenario 2, the results are quite similar. ICE is still able to estimate the correct distribution quite good (0.0430). However, local minima leading to completely wrong results are encountered more frequently. The estimation of *LpMatcher* becomes wider as more incorrectly matched event pairs are used to train the KDE. Yet, the first mode is still calculated fairly good still resulting in a low distance of 0.0405. In contrast, the *lagEM* (0.6499) algorithm nearly completely misses the correct distribution. These results are visible in Figure 4.6.

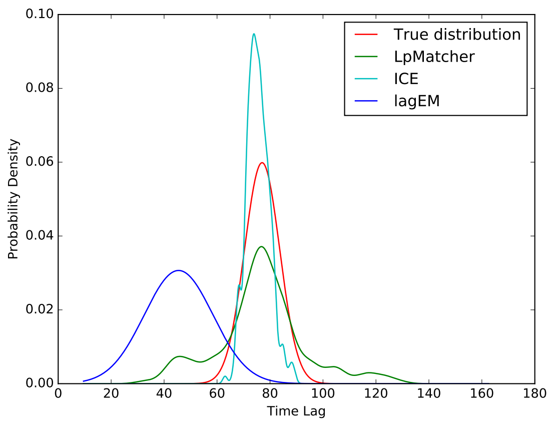
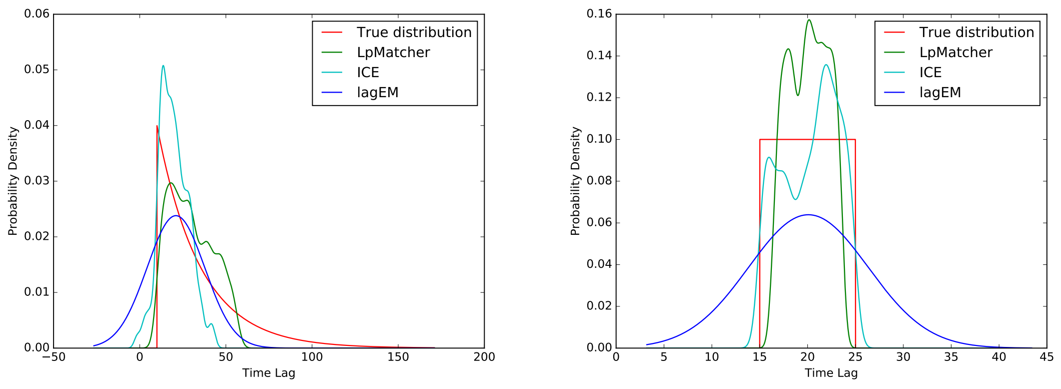


Figure 4.6: Estimated distributions for scenario 3.

Scenario 4
This scenario introduces two additional event types. Furthermore, the temporal lag is not modelled by normal distributions:

> A sequence contains 1000 elements in total. Event A occurs based on an uniform distribution with $\mathcal{U}(30, 50)$. Event B follows after event A with an exponentially distributed time lag with $\beta = 10$ and an offset of 25. Independently of A and B an additional event pair exists. Event C occurs based on an exponential distribution with $\beta = 25$ and an offset of 10. Event D follows after event C with a uniformly distributed time lag $\mathcal{U}(15, 25)$. All event types are lost with a probability of 10%.



(a) Temporal lag between event A and B. (b) Temporal lag between event C and D.

Figure 4.7: Estimated distributions for scenario 4.

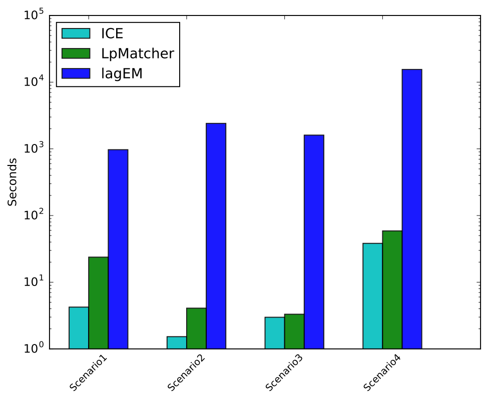
First of all, the hypothesis test using the energy distance is able to detect only correlations between the event types $A \leftrightarrow B$ and $C \leftrightarrow D$. Figure 4.7a shows the estimated distributions between event A and B. Basically, all algorithms are able to estimate the correct distribution quite well. ICE misses the trailing event pairs. Consequently, the distance is quite large with 0.1521. In contrast, *LpMatcher* (0.0351) overestimates the expected mean. The *lagEM* algorithm estimates a similar mean as the *LpMatcher*. But the probability of small or even negative time lags is badly overestimated. This is caused by the inherent assumption that the temporal lag is normally distributed. Nevertheless, the estimate is still better than ICE with 0.0942.

The same behaviour can be observed in Figure 4.7b for the temporal lag between C and D. *lagEM* is able to estimate the correct mean. However, the variance is too big as it is hard to approximate a uniform distribution with a normal distribution. This yields a higher distance of 0.0561. In contrast, both *LpMatcher* (0.0096) and ICE (0.0327) are able to estimate the correct distribution better. The jittery probability density can be explained by the limited number of samples.

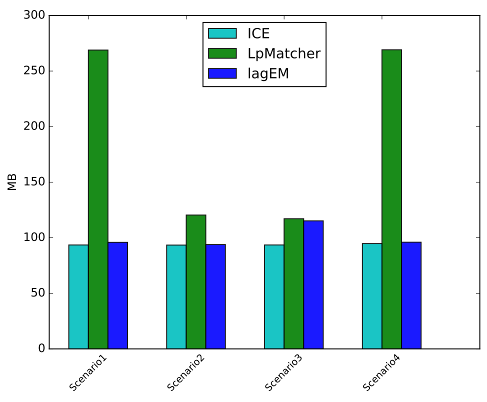
In summary, the *LpMatcher* always yields the estimate with the lowest distance to the true distribution. In all but one case *lagEM* calculates the worst estimate.

Resource Consumption

Besides the quality of the estimation, the resource consumption—namely runtime and memory usage—is an important evaluation factor.



(a) Runtime during different scenarios.



(b) Memory usage during different scenarios.

Figure 4.8: Evaluation of runtime and memory usage of all algorithms during the different scenarios.

As Figure 4.8a shows, *lagEM* has a 50–100 times longer runtime than the other two algorithms. The best runtime is yielded by ICE. Concerning memory consumption, *lagEM* and ICE are very similar and constant for all scenarios. Only, *LpMatcher* requires much more memory in scenarios with many events. This is visible in Figure 4.8b. The complete results are available in Appendix B.

4.3 Symantec Log

After using synthetic data for testing the performance, the next step is to use real world data. Therefore, an event log of *Symantec Endpoint Protection Client* [1] was collected and analysed. The event log is a XML-file with the *Microsoft Event Scheme* containing all Symantec events from 5th March, 2015 to 10th March, 2016. See Listing C.1 for an excerpt of the event log. In total, the log contained 3742 events of 19 different types. Table 4.1 contains all different events with their label, a short description and the number of occurrences. The event description is either taken from the Symantec support website [54] or the description of the event in the log itself.

At first, a manual inspection of the log file was done to search for possibly correlated events. Based on the number of events and event descriptions, the following potential correlations were discovered.

- *2 ↔ 3*: The beginning of an antivirus scan should always be followed by an ending of the scan.
- *51 ↔ 71*: As both events occurred five times, it is reasonable to assume that each detected threat was white listed after a short time.
- *65 ↔ 66*: The events "Scan started" and "Scan stopped" occur roughly equally frequent. Furthermore, a correlation between starting and stopping an antivirus scan sounds plausible.
- *100 ↔ 101*: Again both events occur nearly equally frequent and the connection between "Internet connection lost" and "Internet connection established" may be reasonable.
- *200 ↔ 202*: Both the number of events and the event descriptions "File download finished" and "File installation finished" imply a relationship between these two event ids.
- *34054 ↔ 34057*: Even though the events "Threat protection enabled" and "Manipulation Protection enabled" do not sound correlated, both events occur exactly equally often. Therefore, a correlation may exist.

Next, it was tested which correlations are detected by a hypothesis test using energy distances with a significance level $\alpha = 0.05$. Furthermore, events with a proportion of less than 0.5% are removed as not enough samples are available. In the following all detected correlations are listed and checked for false positive detections. The time lag in seconds is divided by 100 for better readability. Furthermore, the *lagEM* algorithm had to be initialised close to the correct solution as otherwise the support of the estimated normal distribution would be nearly always zero.

[1]For more information see
`https://www.symantec.com/products/threat-protection/endpoint-family/endpoint-protection.`

Event ID	Description	Count
2	Symantec finishes an antivirus scan of the hard drive for malware.	8
3	Symantec starts an antivirus scan of the hard drive for malware.	10
6	Opening a file for malware scanning failed.	92
7	A virus definition with new threat descriptions has been found.	308
12	The configuration of the Symantec scanner has been changed.	3
26	Scheduled scan paused by user.	1
51	Symantec has successfully handled a detected threat.	5
65	Symantec antivirus scan suspended.	36
66	Symantec antivirus scan continued.	38
69	Running malware scan failed due to unknown error.	1
71	A previously detected threat was manually ignored and white listed.	5
100	Antivirus scanner has established an internet connection.	297
101	Antivirus scanner has lost the internet connection.	235
200	The download of a file has finished.	1238
201	The download of a file has failed.	86
202	File installation finished. This event is triggered when a downloaded file was successfully scanned for a threat.	1235
203	File installation failed. This event is triggered when the scan for a threat in a downloaded file failed.	2
34054	Symantec threat protection enabled.	71
34057	Symantec manipulation protection enabled.	71

Table 4.1: List of all event types in the Symantec log file with a short description and the number of occurrences.

2* ↔ *6

This detected correlation is ignored as event *2* does not have enough samples.

3* ↔ *100

This detected correlation is ignored as event *3* does not have enough samples.

7 ↔ 66

It is reasonable to assume that the download of a new virus definition triggers an antivirus scan. Yet, event 7 occurs roughly ten times more than event 66. During a manual inspection multiple possible correlation were detected. Even though the exact correlation was not absolutely clear, most of the manual correlations should be correct.

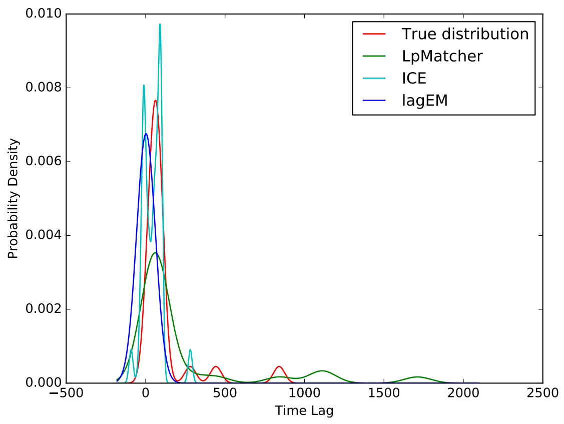


Figure 4.9: Distribution of the temporal lag between event 7 and 66.

Figure 4.9 shows the detected correlations of ICE, *lagEM* and *LpMatcher*. It is visible that all algorithms estimate the main peak of the true distribution quite good. However, no algorithm is able to estimate the true distribution completely correct. The *lagEM* algorithm underestimates the mean, ICE estimates several peaks around the true mean. The best result is calculated by *LpMatcher*, however the estimation includes some outliers with very high time lag. These observations are also visible via the energy distance: The *LpMatcher* yields the best distance with 0.0478, followed by ICE (0.1654) and *lagEM* (0.2669).

Even though a quite strong correlation exists, the correlation might still be a false positive as only roughly 10% of the events with id 7 trigger an event 66.

65 ↔ 66

A correlation between the start and stop of an antivirus scan is quite obvious. A manual inspection of the log file revealed a connection between several event types. At some point an antivirus scan is started (event 3). This scan runs for roughly one hour until it is suspended (event 65). On the next Friday the virus scan is resumed (event 66) for one hour and suspended again. This is repeated until the scan eventually completes (event 2) and a new scan is started one week later. Table C.1 contains a list of those events in correct order. During the inspection 27 correlated event pairs were identified. The remaining 20 events are not correlated leading to roughly ~38% outliers.

Nevertheless, both the ICE and *LpMatcher* are able to exactly detect the correct correlations. The temporal lag is displayed in Figure 4.10.

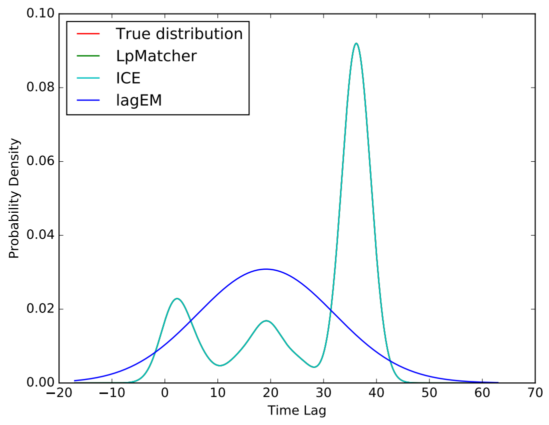


Figure 4.10: Distribution of the temporal lag between event 65 and 66.

Figure 4.10 also shows the estimated time lag by the *lagEM* algorithm. This algorithm computes a time lag distributed by $\mathcal{N}(19.125, 12.9365)$ leading to a distance of 0.1806. However, it is important to note, that the *lagEM* algorithm was initialised with a μ close to the correct solution. Even though the rough distribution is estimated quite well, the estimate by both ICE and *LpMatcher* resembles the true distribution perfectly - yielding a small distance of $3.3182 \cdot 10^{-12}$. This is a very good result as the data set contains roughly 38% outliers.

200 ↔ 202

After successful file download almost always a successful file installation follows. The mere number of events made a complete manual correlation impossible. However, random inspections of the log file revealed that both events happen at the same time. Therefore, the true distribution is assumed to be a δ-function at 0.

Figure 4.11 shows that basically all algorithms estimate the correct distribution. Even though, the mean of the *lagEM* distribution is slightly negative the offset is less than 5 ms. Besides the quality of the estimate, also the calculation speed is relevant for this amount of events. ICE is able to calculate a solution in less than 1 second. In contrast, the *lagEM* algorithm requires more than 91 minutes calculation time. Besides the longest runtime, the *lagEM* algorithm also yields the highest distance (0.8424). In contrast to the other event pairs, ICE (0.0541) outperforms the *LpMatcher* (0.1433).

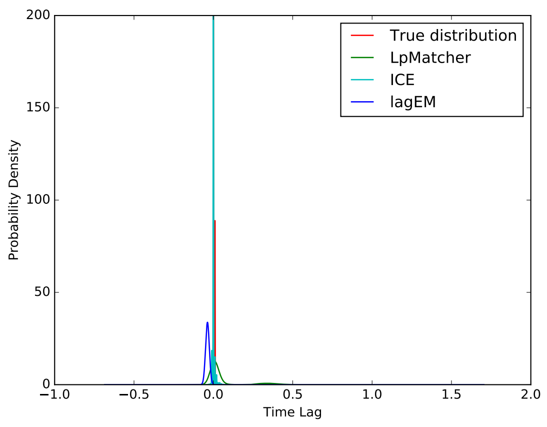


Figure 4.11: Distribution of the temporal lag between event 202 and 200.

34054 ↔ 34057

An inspection of the event time stamps revealed that "Threat Protection enabled" and "Manipulation Protection enabled" always occur at the same time. This implies that both modules are activated at the same time, for example during program start.

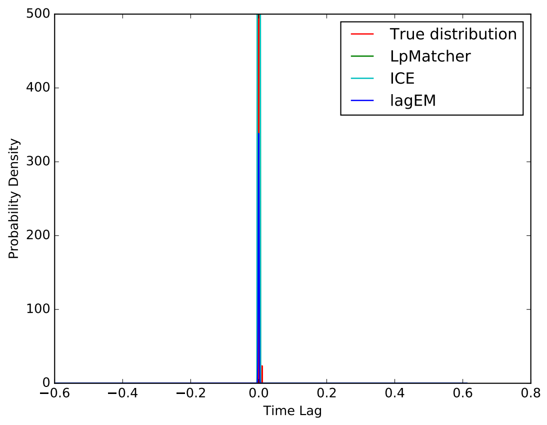


Figure 4.12: Distribution of the temporal lag between event 34054 and 34057.

Figure 4.12 shows the estimated distribution calculated by ICE, *LpMatcher* and *lagEM*. The distributions clearly show that only a time lag close to 0 has a high probability. The *lagEM* algorithm calculated a very thin distribution $\mathcal{N}(-1.4085 \cdot 10^{-4}, 1.1784 \cdot 10^{-3})$. However, even this thin normal distribution differs significantly from the true distribution with a distance of 0.2513. Again, both ICE and *LpMatcher* yield the same distance of 0.0141. So, for this trivial case all algorithms perform equally well.

Finally, all expected correlations that were not discovered have to be evaluated to detect false negatives.

2 ↔ 3

During the manual analysis of events 65 and 66, a correlation between events 2 and 3 was detected. Yet, this correlation was not noticed by the energy distance as the p-value (0.135) was larger than the significance level $\alpha = 0.05$. This may be caused by the very limited number of events.

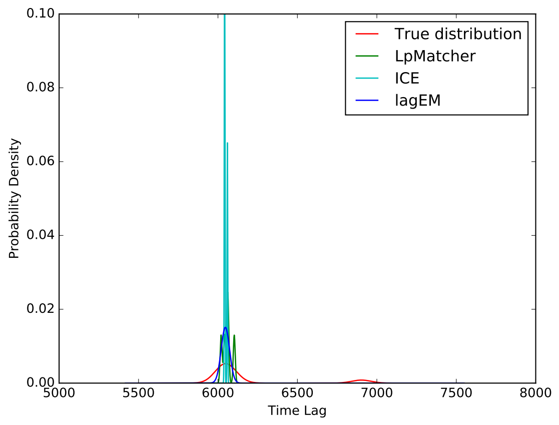


Figure 4.13: Distribution of the temporal lag between event 2 and 3.

As Figure 4.13 shows, all algorithms are capable of detecting the rough distribution of the true time lag with a quite low distance (*LpMatcher* 0.0972, ICE 0.1371, *lagEM* 0.1043). Consequently, this correlation was missed by the energy distance.

51 ↔ 71

A manual analysis revealed that all events of type 51 were fired at the same time. In contrast, the events 71 occur randomly over the course of 9 months. As a result, both event types are truly uncorrelated.

100 ↔ 101

Finally, the correlation between the events 100 and 101 was evaluated. Figure 4.14 shows all detected correlations.

A manual analysis showed that the internet connection was lost for less than 60 seconds in ~24% of all event pairs. However, it is hard to say whether this is a true correlation because the remaining 76% were distributed randomly over a large time span. Furthermore, it is hard to argue that a lost internet connection is always restored after n seconds in general.

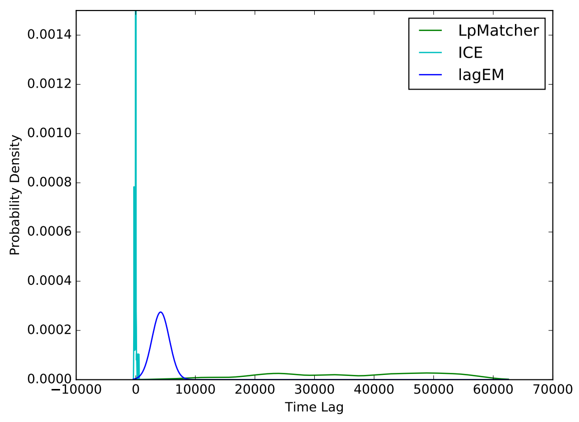


Figure 4.14: Distribution of the temporal lag between event 100 and 101.

At very last, two uncorrelated events were analysed.

201 ↔ 6

Event id 201 and 6 were chosen as their description suggest no correlation and the number of occurrences is nearly the same. The energy distance calculated a *p*-value of 0.997 meaning a correlation is very unlikely.

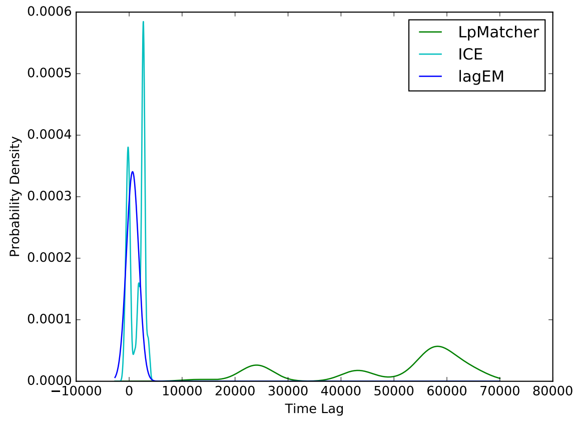


Figure 4.15: Distribution of the temporal lag between event 201 and 6.

Figure 4.15 shows all detected correlations. It is important to note that the *x*-axis covers a range of 92 days. So even though it looks like a time lag centred around 5000 is more likely, this peak still covers a time span of 17 days. Consequently, the calculated distributions clearly differ from the distributions for correlated events.

4.4 Heidelberger Druck - Printer Log

Finally, a log file of an industrial printer is analysed. Therefore, all events of a *Heidelberger Druckmaschienen AG Speedmaster XL 145* between the 30th June, 2015 and 27th August, 2015 were collected. In total, the log file contains 245 925 unique events with 252 different event types.

Due to the huge number of event types, a complete analysis of all detected correlations is infeasible. Furthermore, this would not be reasonable as it is not possible to create a ground truth for the correlations as done for the Symantec log. Instead the sequence is analysed with a given objective:

> In practice printers are quite often manually restarted in case of errors. However, this "*have you tried turning it off and on again*" mentality generally does not solve the underlying problem permanently. Goal of this analysis is to find event types leading to a printer restart.

A printer restart is logged as an event id $0X5001x$ with x being a hexadecimal value. In the given log file the following relevant events can be observed:

Event ID	Description	Count
$0X50017$	Incorrect shutdown of control station computer.	2
$0X50018$	RGP was switched off before IPC.	2
$0X5001E$	Software error - control station computer.	12
$0X5001F$	Software error - SCU.	2

Table 4.2: List of event types connected to a restart with a short description and the number of occurrences.

The events $0X50017$, $0X50018$ and $0X5001F$ are not suited for an analysis as they occur only twice. Consequently, event $0X5001E$ is used. The pure amount of events makes an investigation very time consuming and infeasible on a normal computer. Therefore, the sequence is trimmed to chunks containing events of 30 minutes before a $0X5001x$ event. Those chunks are concatenated to a significantly shortened sequence. Even though this simplification may influence the results, it is reasonable as the event leading to the hard restart should occur close in time to the actual restart. Using the trimmed sequence, all possible correlations are calculated. Figure 4.16 shows all event types directly or indirectly correlated with $0X5001E$.

Even though this network looks quite complicated, it reduced the possible causes for a restart from 248 to 15 event types. Based on this network, the influence of the sequence trimming is explored. As Table 4.3 shows, all events occur 11–26 times in the trimmed sequence. In the original sequence however, event types are observed 12 to 48 963 times. This distortion can be manually corrected by choosing only event types that occur roughly the same time as $0X5001E$—namely $0X200004A$, $0X200012C$, $0X200012D$, $0X200012E$, $0X2000456$ and $0X20005A2$. These event types are also highlighted in Figure 4.16. The correlation of the seven remaining event types can be modelled in a Bayesian network as displayed in Figure 4.17.

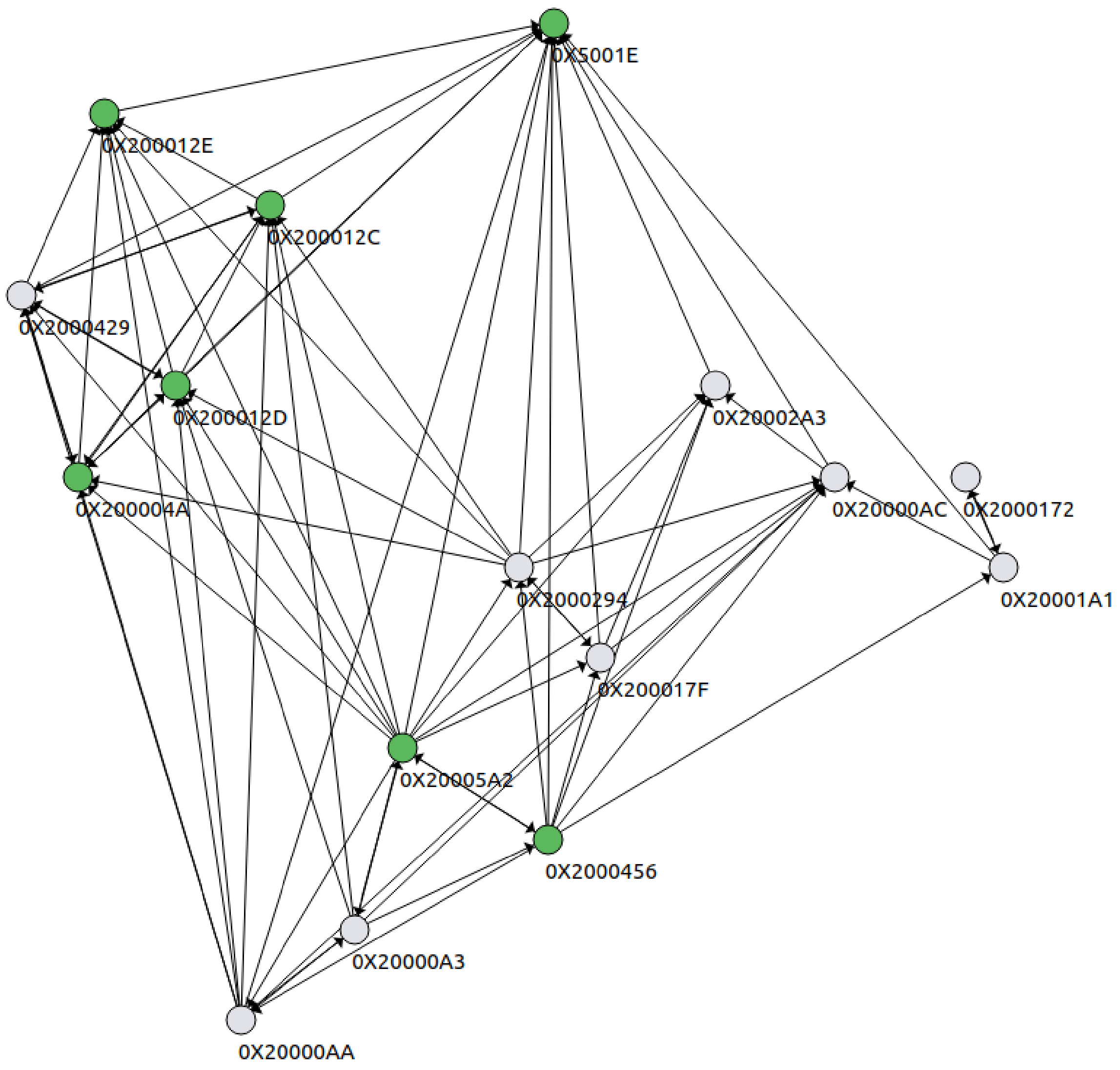


Figure 4.16: Network of events associated with the restart event $0X5001E$. The arrows show a causal dependencies between the connected event types. Highlighted in green are the most promising indicators for a soon-to-be restart.

Event ID	**Complete Count**	**Trimmed Count**
0X200004A	71	11
0X20000A3	48963	18
0X20000AA	214	24
0X20000AC	472	17
0X200012C	76	12
0X200012D	77	12
0X200012E	58	11
0X2000172	2552	17
0X200017F	1517	13
0X20001A1	2464	15
0X2000294	598	12
0X20002A3	4144	13
0X2000429	1141	26
0X2000456	71	13
0X20005A2	86	12
0X5001E	12	12

Table 4.3: Restart correlated event types with the number of occurrences in the original and trimmed sequence.

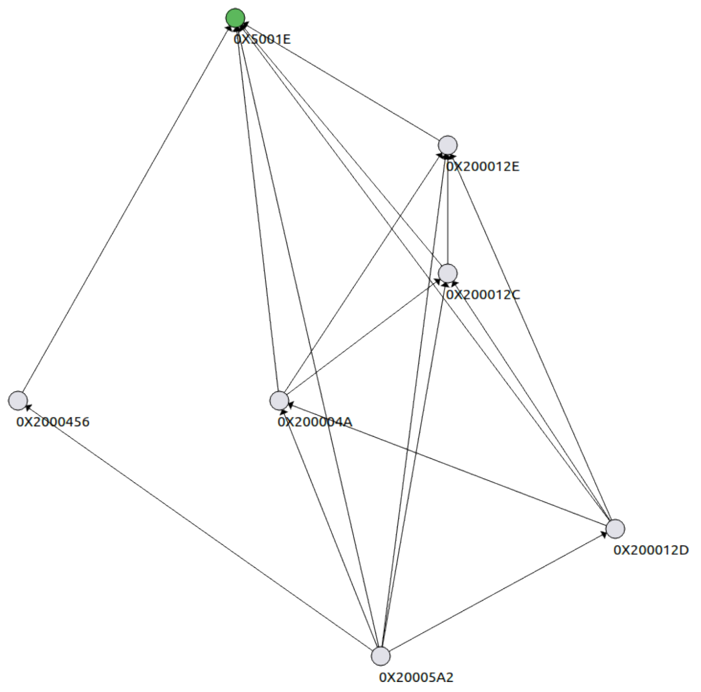


Figure 4.17: Relevant events for restart in a Bayesian network. The actual restart event $0X5001E$ is highlighted in green.

Theoretically, the next step would be verifying the detected correlations against some ground truth. Unfortunately, it was not possible to obtain a ground truth from a domain expert at Heidelberger Druckmaschienen AG. Furthermore, a manual creation of ground truth is not possible as domain knowledge would be required and the log entries are rather cryptic. Therefore, it is simply assumed that all detected correlations are valid and sensible. Based on this assumption, the influence of different event types on $0X5001E$ is evaluated. For better readability let $R = P(0X5001E = ✓)$ be the probability for a restart and $\overline{R} = P(0X5001E = ✗)$ the converse probability. Furthermore, the prefix '2000' is dropped for all other event types, e.g. $0X200012D$ is abbreviated as $0X12D$. All experiments described in the following are also summarised in Table D.1.

At first the probabilities without any evidence is calculated as $P(R) = 0.9895$ and $P(\overline{R}) = 0.0105$. These values correspond with the low occurrence probability of R. By providing only negative evidence, i.e. no other event occurred, the probability of a restart gets even smaller with

$$P(R \mid 0X04A = ✗, 0X12C = ✗, 0X12D = ✗, 0X12E = ✗, 0X456 = ✗, 0X5A2 = ✗) = 0.0001\ .$$

In contrast, if all other events occurred the probability of a restart is 99.99%. Those values are plausible as 84.91% of all triggering events are followed by R. In addition R is preceded by an triggering event in 83.34% of all cases.

Next, it was analysed which event is the strongest indicator for a restart. Therefore, only one specific event is used as evidence. As Figure 4.18 shows, the occurrence probability of R is very low. This result can be explained as the average probability of an event to occur is 0.0152. Accordingly, it is much more likely that all missing events did not occur resulting in a small $P(R)$. In a second test, all missing events are provided as negative evidence. As expected, the occurrence of R becomes even more unlikely. Consequently, a single observed event is not sufficient for predicting R. Instead, the combination of several events should be considered.

In the next step, two event types are combined to calculate the probability of R. Figure 4.19 shows some exemplary combinations of events, for all values see again Table D.1. The results from this test coincide with the results in Figure 4.18. The combinations of the two events with the lowest probability yields a very unlikely restart $P(R \mid 0X456 = ✓, 0X12D = ✓) = 0.0169$. Similarly weak is the correlation $P(R \mid 0X456 = ✓, 0X12E = ✓)$. In contrast, $0X12C$ combined with $0X5A2$—both being the best single indicator for a restart—results in a high probability $P(R \mid 0X12C = ✓, 0X5A2 = ✓) = 0.8918$. Also the events $0X5A2$ and $0X04A$ are useful indicators. Nearly all remaining event pairs yield a probability in $[0.1, 0.4]$ and hence they are rather bad indicators for a restart.

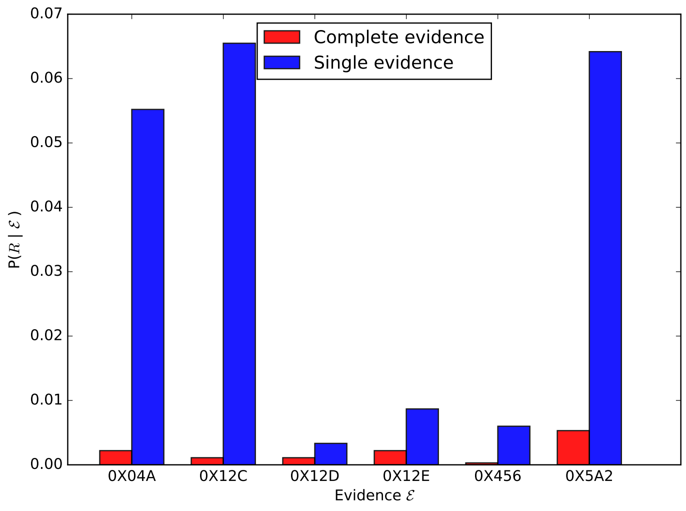


Figure 4.18: Probability for a restart given different event types as evidence. For single evidence only the given event type was provided as positive evidence. For complete evidence the given event type was provided as positive evidence and all remaining event types as negative evidence.

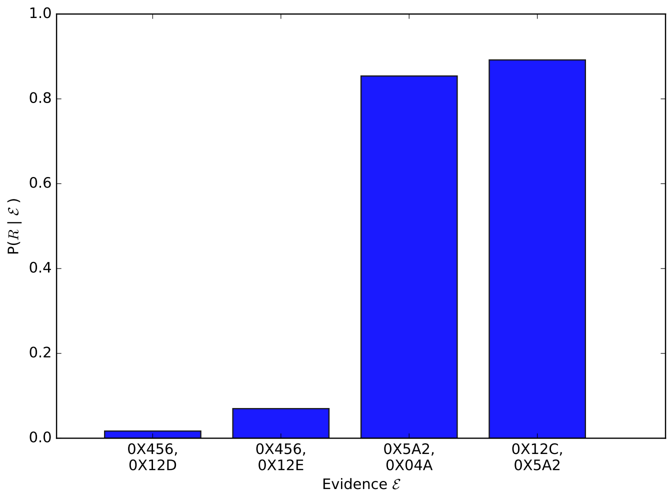


Figure 4.19: Probability for a restart given different combinations of two event types as positive evidence. All other events are not included in the evidence.

Chapter 5

Conclusion

This work presented a new data-driven approach for event correlation. Therefore, a general framework was introduced that only utilises an event time stamp and label. This framework is able to automatically transform a sequence into Bayesian network without any additional input as described in Section 3.1 and 3.4. It uses a preliminary test for goodness of correlation to determine promising event pairs before applying a time consuming matching. These promising pairs can be processed by ICE, *LpMatcher* or an already existing approach like *lagEM*. By examining only pair-wise event types, this framework is highly applicable in big data as all event pairs can be processed in parallel.

For calculating the temporal lag distribution between two event types, two new algorithms have been proposed. Both approaches do not require any a priori information, e.g. a good initialisation or domain knowledge. Furthermore, no inherent assumptions about the true distribution of the temporal lag is included making both algorithms highly flexible.

ICE, an adaptation of ICP to event sequences, is a simple algorithm. While obtaining results similar to state-of-the-art approaches, the runtime is significantly less. Instead of a few hours calculation time, results are available within seconds. This makes the algorithm optimal to get a rough first impression of the temporal lag for preliminary analysis.

The second approach formulates event correlation as a linear minimisation called *LpMatcher*. It was shown that using several approximations an originally NP-hard integer quadratic optimisation problem is efficiently solvable with linear programming. Furthermore, it was proven that these approximations introduce only a small additional error. Even though this algorithm is slower than ICE, it still outperforms existing approaches in terms of runtime. Furthermore, nearly always the goodness of estimation is better than ICE and *lagEM*. In combination with the diverse and expressive possibilities to incorporate optional domain knowledge, *LpMatcher* is a powerful matcher for event correlation.

Tests with simulated data and two real data sets showed the power of the presented methods. Using the Symantec log, several diverse and complicated temporal lag distributions have been estimated and analysed in Section 4.3. This data set demonstrated the advantage of eliminating assumptions about the true distribution from the event correlation process as both ICE and *LpMatcher* have much more precise estimates than *lagEM*.

In contrast, the Heidelberger Druckmaschienen AG log, presented in Section 4.4, showed the power of the energy distance as a preselection criterion. From over 250 event types, six have been identified as causes for a restart. Furthermore, the complex dependency graph of all event types was successfully transformed into a Bayesian network allowing reasoning about the restart probability. During this analysis, the best indicators for a restart have been identified without any domain knowledge. In addition, this data set showed that the methods work with very few samples as each event type occurred less than 25 times.

In the future, the construction of the Bayesian network should be improved. For simplicity, events were modelled as discrete random variables and the actual temporal lag was ignored. It would be desirable to model all events as continuous random variables with their occurrence time stamp as the realisation of it. Therefore, the Bayesian network has to be extended as described in [55]. ICP is a well researched algorithm with many improvements and extensions. These improvements may be transferred to ICE. Furthermore, ICE could be improved to make it more outlier resistant by using a different cost function or a better *data* point cloud filtering. Finally, the *LpMatcher* uses much memory for sequences with a few thousand events. Further research could either make the problem even simpler or use a better, more efficient solver for linear programming.

Bibliography

[1] C. Zeng, L. Tang, T. Li, L. Shwartz, and G. Y. Grabarnik, “Mining temporal lag from fluctuating events for correlation and root cause analysis,” *Proceedings of the 10th International Conference on Network and Service Management, CNSM 2014*, pp. 19–27, 2015.

[2] S. Benferhat, T. Kenaza, and A. Mokhtari, “A Naive Bayes Approach for Detecting Coordinated Attacks,” *32nd Annual IEEE International Computer Software and Applications (COMPSAC)*, pp. 704–709, 2008.

[3] K. Houck, S. Calo, and A. Finkel, “Towards a Practical Alarm Correlation System,” in *Integrated Network Management IV*. Springer US, 1995, pp. 226–237.

[4] H.-J. Kettschau, S. Bruck, and P. Schefczik, “LUCAS - an Expert System for Intelligent Fault Management and Alarm Correlation,” in *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*. IEEE, 2002, pp. 903–905.

[5] R. Gopal, “Layered model for supporting fault isolation and recovery,” in *Network Operations and Management Symposium (NOMS)*. IEEE, 2000, pp. 729–742.

[6] W. Zhou, T. Li, L. Shwartz, and G. Y. Grabarnik, “Recommending Ticket Resolution using Feature Adaptation,” in *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, 2015, pp. 15–21.

[7] F. J. Molinero Velasco, “A Bayesian Network approach to diagnosing the root cause of failure from Trouble Tickets,” *Artificial Intelligence Research*, vol. 1, no. 2, pp. 75–85, 2012.

[8] G. Jakobson and M. D. Weissman, “Alarm Correlation,” *IEEE Network: The Magazine of Global Internetworking*, vol. 7, no. 6, pp. 52–59, 1993.

[9] H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of Frequent Episodes in Event Sequences,” *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 259–289, 1997.

[10] K. Bouandas and A. Osmani, “Mining Association Rules in Temporal Sequences,” *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 610–615, 2007.

[11] B. Efron, "Bayes' Theorem in the 21st Century," *Science*, vol. 340, no. 6137, pp. 1177–1178, 2013.

[12] A. A. Dempster, N. N. Laird, and D. D. B. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society Series B Methodological*, vol. 39, no. 1, pp. 1–38, 1977.

[13] E. L. Lehmann and J. P. Romano, *Testing Statistical Hypotheses*, 3rd ed. Springer New York, 2008.

[14] Student, "The Probable Error of a Mean," *Biometrika*, vol. 6, no. 1, pp. 1–25, 1908.

[15] The Pennsylvania State University, "Hypothesis Testing," 2017. [Online]. Available: https://onlinecourses.science.psu.edu/statprogram/node/138

[16] M. Rosenblatt, "Remarks on Some Nonparametric Estimates of a Density Function," *The Annals of Mathematical Statistics*, vol. 27, no. 3, pp. 832–837, 1956.

[17] J. Babaud, A. P. Witkin, M. Baudin, and R. O. Duda, "Uniqueness of the Gaussian Kernel for Scale-Space Filtering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 1, pp. 26–33, 1986.

[18] D. W. Scott, *Multivariate Density Estimation : Theory, Practice, and Visualization*. New York: Wiley-Interscience, 1992.

[19] Q. Yao and H. Tong, "Cross-Validatory Bandwidth Selections For Regression Estimation Based on Dependent Data," *Journal of statistical planning and inference*, vol. 68, no. 2, pp. 387–415, 1998.

[20] V. C. Raykar and R. Duraiswami, "Very fast optimal bandwidth selection for univariate kernel density estimation," *Technical Reports from UMIACS UMIACS-TR-2005-73*, pp. 522–526, 2006.

[21] A. W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Biometrika*, vol. 71, no. 2, pp. 353–360, 1984.

[22] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 347-352, pp. 240–242, 1895.

[23] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson Correlation Coefficient," in *Noise Reduction in Speech Processing*. Springer Berlin Heidelberg, 2009, pp. 1–4.

[24] E. Garcia, "A Tutorial on Correlation Coefficients," p. 1, 2010. [Online]. Available: http://web.simmons.edu/~benoit/lis642/a-tutorial-on-correlation-coefficients.pdf

[25] C. Shannon, "Prediction and Entropy of Printed English," *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[26] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, Inc., 2006.

[27] M. L. Rizzo and G. J. Székely, “Energy distance,” *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 8, no. 1, pp. 27–38, 2016.

[28] P. J. Besl and N. D. McKay, “A Method for Registration of 3-D Shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[29] P. Glira, “Iterative Closest Point (ICP),” 2016. [Online]. Available: https://www.youtube.com/watch?v=uzOCS_gdZuM

[30] F. Pomerleau, F. Colas, and R. Siegwart, “A Review of Point Cloud Registration Algorithms for Mobile Robotics,” *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[31] E. Ezra, M. Sharir, and A. Efrat, “On the performance of the ICP algorithm,” *Computational Geometry*, vol. 41, no. 1, pp. 77–93, 2008.

[32] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, “The Trimmed Iterative Closest Point algorithm,” in *16th International Conference on Pattern Recognition*, vol. 3. IEEE, 2002, pp. 545–548.

[33] J. M. Phillips, R. Liu, and C. Tomasi, “Outlier robust ICP for minimizing fractional RMSD,” *Proceedings 6th International Conference on 3-D Digital Imaging and Modeling (3DIM)*, pp. 427–434, 2007.

[34] C. B. Do and S. Batzoglou, “What is the expectation maximization algorithm?” *Nature Biotechnology*, vol. 26, no. 8, pp. 897–899, 2008.

[35] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[36] R. B. Rusu, N. Blodow, and M. Beetz, “Fast Point Feature Histograms (FPFH) for 3D registration,” *IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[37] S. So, “Why is the sample variance a biased estimator?” pp. 1–4, 2008.

[38] W. J. Reichmann, *Use And Abuse of Statistics*. Oxford University Press, 1961.

[39] K. G. Murty and S. N. Kabadi, “Some NP-complete problems in quadratic and nonlinear programming,” *Mathematical Programming*, vol. 39, no. 2, pp. 117–129, 1987.

[40] S. Dragomir, “A survey on Cauchy-Bunyakovsky-Schwarz type discrete inequalities.” *Journal of Inequalities in Pure and Applied Mathematics*, vol. 4, no. 3, pp. 1–142, 2003.

[41] S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*. McGraw-Hill, 2006.

[42] C. H. Papadimitriou, *Computational Complexity*. Addison Wesley Longman, 1994.

[43] J. A. Nelder and R. Mead, “A simplex method for function minimization,” *Computer Journal*, vol. 7, pp. 308–313, 1965.

[44] D. Spielman and S. Teng, "Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time," *Journal of the Association for Computing Machinery*, vol. 51, no. 3, pp. 385–463, 2004.

[45] S. Arora, B. Bollobás, and L. Lovász, "Proving Integrality Gaps without Knowing the Linear Program," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2002, pp. 313–322.

[46] G. Sierksma, *Linear and Integer Programming: Theory and Practice, Second Edition*, ser. Advances in Applied Mathematics. Taylor & Francis, 2001.

[47] N. Friedman, D. Geiger, and M. Goldszmit, "Bayesian Network Classifiers," *Machine Learning*, vol. 29, no. 2, pp. 131–163, 1997.

[48] D. Margaritis, "Learning Bayesian Network Model Structure from Data," Doctoral Dissertation, Carnegie Mellon University, 2003.

[49] J. N. K. Rao and A. J. Scott, "The Analysis of Categorical Data From Complex Sample Surveys: Chi-Squared Tests for Goodness of Fit and Independence in Two-Way Tables," *Journal of the American Statistical Association*, vol. 76, no. 374, pp. 221–230, 1981.

[50] R. Hassin and S. Rubinstein, "Approximations for the maximum acyclic subgraph problem," *Information Processing Letters*, vol. 51, no. 3, pp. 133–140, 1994.

[51] G. F. Cooper, "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks," *Artificial Intelligence*, vol. 42, no. 2-3, pp. 393–405, 1990.

[52] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Pearson Education, 2010.

[53] J. Blömer and K. Bujna, *Adaptive Seeding for Gaussian Mixture Models*. Cham: Springer International Publishing, 2016, pp. 296–308.

[54] Symantec Corporation, "Symantec Endpoint Protection 12.1.x event log entries," 2013. [Online]. Available: https://support.symantec.com/en_US/article.TECH186925.html

[55] B. R. Cobb and P. P. Shenoy, "Operations for inference in continuous Bayesian networks with linear deterministic variables," *International Journal of Approximate Reasoning*, vol. 42, no. 1-2, pp. 21–36, 2006.

Appendix A

Performance Evaluation lagEM

Parameter	Lower bound	Upper bound
μ	0	42
μ	42	62
μ	62	92
μ	92	112
μ	112	200
σ	1	3.6
σ	3.6	5.6
σ	5.6	7.6
σ	7.6	9.6
σ	9.6	25

Table A.1: Boundaries for the uniform distributions used to initialise μ and σ. The intervals were chosen such that they are symmetric around the true parameters $\mu \approx 77$ and $\sigma \approx 6.6$.

Appendix B

Synthetic Sequences

Scenario	**Algorithm**	**Runtime**	**Memory**
1	ICE	00:04.249	93.531250
		00:04.247	93.527344
		00:04.204	93.460938
1	*LpMatcher*	00:23.136	268.882812
		00:23.516	268.925781
		00:24.657	268.796875
1	*lagEM*	16:02.331	95.910156
		16:30.447	95.621094
		16:01.135	96.066406
2	ICE	00:01.402	93.261719
		00:01.801	93.628906
		00:01.374	93.324219
2	*LpMatcher*	00:03.970	120.691406
		00:04.188	120.328125
		00:04.087	120.273438
2	*lagEM*	40:22.599	94.105469
		39:59.316	93.628906
		40:01.812	93.912349

Table B.1: Resource consumption of all algorithms for scenario 1 and 2. The last two columns contain the runtime in minutes and memory usage in megabytes.

Scenario	**Algorithm**	**Runtime**	**Memory**
3	ICE	00:03.051	93.929688
		00:03.354	93.207031
		00:02.538	93.414062
3	*LpMatcher*	00:03.191	117.078125
		00:03.279	116.039062
		00:03.460	118.242188
3	*lagEM*	26:24.225	115.679688
		26:46.523	115.762610
		27:06.666	114.210671
4	ICE	00:14,458	94.289062
		00:12,073	94.667969
		00:11,715	95.312500
4	*LpMatcher*	00:57,757	268.878906
		00:59,488	269.621094
		00:59,165	268.906250
4	*lagEM*	252:15.917	95.792969
		248:09.102	96.496094
		275:10.983	95.667969

Table B.2: Resource consumption of all algorithms for scenario 3 and 4. The last two columns contain the runtime in minutes and memory usage in megabytes.

Appendix C

Symantec - Antivirus Log File

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Events>
  <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'>
    <System>
      <Provider Name='Symantec Endpoint Protection Client'/>
      <EventID Qualifiers='0'>200</EventID>
      <Level>4</Level>
      <Task>2</Task>
      <Keywords>0x80000000000000</Keywords>
      <TimeCreated SystemTime='2015-11-27T15:01:19.000000000Z'/>
      <EventRecordID>2712</EventRecordID>
      <Channel>Symantec Endpoint Protection Client</Channel>
      <Security/>
    </System>
    <EventData>
      <Data>
        Inhalte wurden auf den Client heruntergeladen

        Produkt: SEPC Virus Definitions Win64 (x64) 12.1 RU6
        Version: MicroDefsB.CurDefs
        Sprache: SymAllLanguages
        Sequenz: 151126035
        Veroeffentlichungsdatum: Donnerstag, 26. November 2015
        Revision: 035
        Quelle: Symantec Endpoint Protection Manager
        Groesse: 587824 Byte
      </Data>
    </EventData>
  </Event>
  ...
</Events>
```

Listing C.1: Excerpt of the Symantec XML log file. The listing shows all information for a single event. Only lines marked in red are extracted to obtain an event id and event time stamp. All other information is ignored.

Id	Time stamp	Id	Time stamp	Id	Time stamp
3	2015-03-06 12:00:00	65	2015-07-17 19:51:23	3	2015-11-13 13:06:51
66	2015-03-27 11:21:46	66	2015-07-24 12:00:02	65	2015-11-13 13:10:15
66	2015-03-27 12:02:05	65	2015-07-24 13:00:00	66	2015-11-20 12:00:00
66	2015-04-10 12:00:03	66	2015-07-31 12:00:02	65	2015-11-20 13:00:19
2	2015-04-10 12:04:02	65	2015-07-31 13:00:00	66	2015-11-27 12:00:02
3	2015-04-17 12:23:03	66	2015-08-07 12:00:03	65	2015-11-27 13:00:00
65	2015-04-17 13:00:00	65	2015-08-07 13:00:00	66	2015-12-04 12:00:04
66	2015-04-24 12:00:01	66	2015-08-24 10:22:12	65	2015-12-04 13:00:56
65	2015-04-24 13:00:00	65	2015-08-24 10:45:19	66	2015-12-11 12:00:02
66	2015-05-04 09:17:57	66	2015-08-28 22:23:41	2	2015-12-11 12:11:28
3	2015-05-08 12:00:00	65	2015-08-28 22:28:28	3	2015-12-18 12:00:01
65	2015-05-08 13:01:20	66	2015-09-04 12:00:00	65	2015-12-18 13:00:00
66	2015-05-18 09:03:32	65	2015-09-04 13:00:52	66	2016-01-11 09:55:09
65	2015-05-18 09:05:35	66	2015-09-14 09:50:52	65	2016-01-11 10:25:28
66	2015-05-22 12:00:04	65	2015-09-14 10:32:52	66	2016-01-18 09:05:55
65	2015-05-22 13:00:00	66	2015-09-18 12:00:04	65	2016-01-18 09:16:51
66	2015-05-29 12:00:01	2	2015-09-18 12:47:17	66	2016-01-22 12:00:04
65	2015-05-29 13:01:00	3	2015-09-25 12:00:00	65	2016-01-22 13:00:03
66	2015-06-05 22:38:38	65	2015-09-25 13:00:00	66	2016-01-29 12:00:03
65	2015-06-05 22:42:00	66	2015-10-02 12:00:02	65	2016-01-29 13:00:00
66	2015-06-12 12:26:07	65	2015-10-02 13:00:00	66	2016-02-05 12:00:04
65	2015-06-12 13:00:00	66	2015-10-09 12:00:03	2	2016-02-05 12:09:18
66	2015-06-19 12:00:02	65	2015-10-09 13:03:01	3	2016-02-12 12:00:03
65	2015-06-19 13:00:00	66	2015-10-16 12:00:03	65	2016-02-12 13:00:00
66	2015-06-29 09:50:12	2	2015-10-16 12:42:00	66	2016-02-19 12:00:01
65	2015-06-29 10:22:23	3	2015-10-23 12:00:02	65	2016-02-19 13:00:00
66	2015-07-03 12:00:02	65	2015-10-23 13:00:00	66	2016-02-26 12:00:03
2	2015-07-03 12:44:12	66	2015-10-30 12:00:02	2	2016-02-26 12:45:20
3	2015-07-10 14:15:01	65	2015-10-30 13:00:00	3	2016-03-04 12:00:03
65	2015-07-10 14:16:12	66	2015-11-06 12:00:00	65	2016-03-04 13:00:00
66	2015-07-17 19:48:21	2	2015-11-06 12:40:58		

Table C.1: List of all events emitted by antivirus scan in correct order. This pattern can be explained as the antivirus scan was scheduled to run for one hour a week. So, if the virus scan is started (event 3) but has not finished within one hour, it will be suspended (event 65) until next week. Then the virus scan is resumed (event 66) for one hour and suspended again until eventually the scan completes (event 2).

Appendix D

Heidelberger Druck - Printer Log

```
{
  "userData":0,
  "yearmonthID":"2015_06",
  "netID":0,
  "sym":0,
  "yearweekID":"2015_27",
  "function":0,
  "ContextID":"0X10010000",
  "subIndex2":0,
  "i_measDev":0,
  "modulID":0,
  "timestamp":"2015-06-30T08:04:04.573+0200",
  "Status":"SET",
  "subIndex":0,
  "poscode":1,
  "subIndex1":0,
  "Text":"Lack of dampening solution",
  "FailureID":"0X20001E2",
  "unitindex":0,
  "equipmentID":0,
  "functionID":0,
  "l_OKZ_direkt":0,
  "machineNumber":"XXX",
  "TimeOfOccurence":"2015-06-30T08:04:04.573+0200",
  "locationID":1,
  "yeardayID":"2015_181",
  "sublocation":0
}
```

Listing D.1: Excerpt of the Heidelberger Druck log file. The listing shows all information for a single event as a JSON. Only the lines marked in red are extracted to obtain an event id and event time stamp. All other information is ignored.

Evidence						Probability	
0X200004A	0X200012C	0X200012D	0X200012E	0X2000456	0X20005A2	$P(\overline{R})$	$P(R)$
-	-	-	-	-	-	0.9895	0.0105
✗	✗	✗	✗	✗	✗	0.9999	0.0001
✓	✓	✓	✓	✓	✓	0.0001	0.9999
✓	-	-	-	-	-	0.9448	0.0552
-	✓	-	-	-	-	0.9345	0.0655
-	-	✓	-	-	-	0.9967	0.0033
-	-	-	✓	-	-	0.9913	0.0087
-	-	-	-	✓	-	0.9940	0.0060
-	-	-	-	-	✓	0.9358	0.0642
✓	✗	✗	✗	✗	✗	0.9978	0.0022
✗	✓	✗	✗	✗	✗	0.9989	0.0011
✗	✗	✓	✗	✗	✗	0.9989	0.0011
✗	✗	✗	✓	✗	✗	0.9978	0.0022
✗	✗	✗	✗	✓	✗	0.9997	0.0003
✗	✗	✗	✗	✗	✓	0.9947	0.0053
✓	✓	-	-	-	-	0.1771	0.8229
✓	-	✓	-	-	-	0.7667	0.2333
✓	-	-	✓	-	-	0.7354	0.2646
✓	-	-	-	✓	-	0.6624	0.3376
✓	-	-	-	-	✓	0.1461	0.8539
-	✓	✓	-	-	-	0.7229	0.2771
-	✓	-	✓	-	-	0.8589	0.1411
-	✓	-	-	✓	-	0.6268	0.3732
-	✓	-	-	-	✓	0.1082	0.8918
-	-	✓	✓	-	-	0.8978	0.1022
-	-	✓	-	✓	-	0.9831	0.0169
-	-	✓	-	-	✓	0.4927	0.5073
-	-	-	✓	✓	-	0.9302	0.0698
-	-	-	✓	-	✓	0.2885	0.7115
-	-	-	-	✓	✓	0.9250	0.0750

Table D.1: Probability of the restart event based on various evidences as calculated by the Bayesian network displayed in Figure 4.17. ✗ means that the according event did not happen, ✓ that the according event happened and '- ' that it was not observed meaning it could have occurred or not.