POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Fault analysis of a complex electrical distribution system with Bayesian networks and Markov chains

Relatore: Prof. Francesco Amigoni
Correlatore: Dr. Mirjana Mazuran

Tesi di Laurea Magistrale di:
Alessandro Pozzi, matricola 852358
Lorenzo Costantini, matricola 852599

Anno Accademico 2016-2017

Sommario

Non è insolito che aziende o grandi organizzazioni abbiano a che fare con sistemi complessi, per esempio con infrastrutture composte da una estesa rete di componenti che interagiscono l'uno con l'altro. Più il sistema si sviluppa con il tempo, più diventa difficile tener traccia delle correlazioni tra i componenti e mantenere un modello complessivo del suo funzionamento. Queste correlazioni diventano rilevanti quando si presentano delle anomalie, in quanto guasti e comportamenti inaspettati di alcuni componenti possono influenzarne altri, causando una propagazione di irregolarità. In genere, in queste situazioni, vi è la disponibilità di una grande quantità di dati, che possono essere usati per apprendere un modello del sistema senza l'ausilio di esperti. In questo lavoro, ci concentriamo su una infrastruttura elettrica complessa e distribuita e proponiamo metodi per costruire modelli delle correlazioni fra i componenti elettrici a partire dai dati. In particolare, i modelli rappresentano cosa succede prima o dopo le irregolarità riportate dal sistema. Consideriamo due classi di modelli, reti Bayesiane e catene di Markov, e, nel corso della tesi, discuteremo della loro efficacia, implementando miglioramenti pensati per il nostro contesto e mostrando diverse regolarità fra i componenti della specifica infrastruttura elettrica considerata che sono stati in grado di identificare.

Abstract

In several contexts, companies and institutions may have to deal with complex systems, with an infrastructure composed of an extended network of components that interact with each other. The more the system grows over time, the harder becomes to keep track of the correlations among the components and of the overall model of the system. Knowing these correlations is relevant when some anomalies occur, because failures and unexpected behaviours of some components can affect other components, causing the propagation of irregularities. These situations are often characterized by the presence of an high quantity of data, which can be used to directly learn a model of the system without the help of an human expert. In this work, we focus on a complex and distributed electrical infrastructure and we build data-driven models with the aim of identifying correlations between electrical components that represent what happens before or after some irregularities reported by the system. We consider two types of models, namely Bayesian networks and Markov chains, and through the thesis we will discuss their effectiveness, propose context-specific improvements, and show recurrent patterns they were able to find between the components of the electrical infrastructure we consider.

Alle nostre famiglie.

Contents

List of Figures

Chapter 1

Introduction

In several contexts, companies and institutions have to deal with large technical infrastructures, often composed of multiple subsystems that interact and influence each other in a network of rich, complex interconnections. Examples include prominently the electrical infrastructures and the information infrastructures. When the infrastructure grows over time and new subsystems are added and integrated with legacy units, it becomes increasingly harder to keep track of the correlations between the subsystems in a coherent and updated model of the infrastructure. The importance of these correlations is remarkable from the point of view of the reliability and availability of the infrastructure. For example, knowing which components will be likely affected by the failure of other components can save time and resources. A manual analysis of the relationships between these components would result in an unreadable breakdown structure, extremely hard to manage. An automated analysis based on data collected during the operation of the infrastructure, instead, could allow to identify correlations between components. It is especially important to find out and analyze irregularities that may propagate to the rest of the system, often called anomalies. All the methods which try to automatically identify anomalies from data are part of a field of study called anomaly detection [4].

In this thesis, we focus on a complex electrical infrastructure with the aim of identifying correlations between electrical components related to the detection of anomalies. These components, which are called *devices*, regularly report events in a log. Our objective is to generate models based on Artificial Intelligence approaches to analyze the correlations between the devices by observing the events reported on the log. In particular, we will focus on Bayesian networks and Markov chains. Bayesian networks are a probabilistic graphical model [2] that allows us to identify the relationships among some variables; Markov chains are stochastic models [3] that we will use to capture the temporal sequentiality of events.
Throughout the thesis we analyze the potential and the difficulties in the use of these models in our applicative context, such as the problem of identifying a temporal period that determines the boundary in which two events in the log can be considered to be correlated, or the issue of duplicated events in the training phase of the model generation. We pay particular attention to the variable selection criteria, which are essential to determine which correlations we could find. We also propose some graphical techniques that can be applied after the model creation in order to improve the immediacy and clarity of its fruition by an human operator. Moreover, through this thesis we will show the results of our models to demonstrate their effectiveness in finding frequent patterns in the sequences of events in the log which are related to faults.

In the literature, anomaly detection has been applied to a great variety of contexts, from intrusion detection in security systems [5] to electrical power systems [6], with techniques that go from graph matching [7] to deep neural networks [8]. Probabilistic graphical models like Bayesian networks [2] are not often used in fault analysis, but applications are not lacking. For example, they have been utilized to predict the probability of failures based on live data [9], and specialized models to be used in failure prediction have been discussed in [10].

Markov chains have been employed in failure detection [11] and for detecting anomalies in event logs [12]. However, differently from most approaches to anomaly detection, our models do not learn the nominal behaviour of the system and use it to predict unusual anomalous deviations, instead they capture what happens before or after a set of some anomalous events.

The thesis is structured as follows:

- Chapter 2 (**State of the art**). In this chapter we present the state of the art on the topics relevant to our work, and we illustrate in detail the modelling techniques we are going to use.

- Chapter 3 (**Problem setting**). In this chapter we describe and formalize our problem and we justify the choices of the techniques we employ to solve it.

- Chapter 4 (**The Bayesian network model**). In this chapter we show the process followed to generate a Bayesian network that represents correlations between components and we discuss its meaning and effectiveness.

- Chapter 5 (**The Markov chain model**). In this chapter we present the steps needed to build a Markov chain and we interpret it, with a structure that follows the one of Chapter 4.

- Chapter 6 (**Examples of use**). In this chapter show how to use the models generated, according to the different assumptions on the problem discussed in Chapters 3, 4, and 5.

- Chapter 7 (**Conclusion and future development**). In this chapter we briefly summarize the thesis and discuss possible future developments that could improve our work.

Chapter 2

State of the art

In this section we present the state of the art of the techniques relevant to our thesis. In Section 2.1 we show the main approaches to anomaly detection, a problem that shares some similarities with the one that we tackle in this work. In Section 2.2 we present the Bayesian network model and in Section 2.3 the Markov chain model.

2.1 Approaches to anomaly detection

Anomaly detection consists in identifying non standard behaviour of systems. The usual approach to perform this task consists in the construction of a model of the system's nominal behaviour. Consequently, this model allows also to detect when the system does not follow the normal behaviour. Anomaly detection is often a very important task because the anomalies discovered from the application of a model or from data analysis may be useful for many application domain contexts. The nominal model can be generated with the help of an expert who already knows the conditions under which the system can be said to follow a regular functioning, or alternatively the data collected from the system can help to discover patterns that determine the correct and incorrect system behaviour. Outlier detection is sometimes used as a synonym for anomaly detection in data mining applications that aim to discover data points that are significantly different from the norm. We can distinguish two main paradigms to anomaly detection: model-based and data-driven.

Model-based approaches require some degree of prior knowledge about the system. A model can be represented analytically or qualitatively and a discussion on these two classes of model-based approaches is given in [13] and [14]. Analytical methods, like parameter estimation, parity space estimation, observer-based estimation, have in common the creation of the residual signal, which is the difference between the measurement of the system and its estimation. The task of these mathematical methods is to design a residual signal which is zero or near to zero when no anomaly occurs while it should differ distinctly from zero otherwise [15]. Many qualitative model based techniques have been applied to Fault Detection, Isolation and Recovery (FDIR) problems: fault trees and statecharts [16], fault propagation models [17], directed graphs [18], expert systems [19] and many other variants. A common strategy of these qualitative methods is to model the behaviour of the system with abstraction hierarchies or causal relationships. The model built in this way should capture the regularity of the behaviour of the system. Model-based techniques have been successfully applied also for the diagnosis of electrical power systems: a probabilistic model-based approach to perform this task, which exploits Bayesian networks and arithmetic circuits, has been proposed in [20]. However, it is not always possible to use a model-based technique. Sometimes there is no human expert who really knows how the entire system works; if there is one, modelling may be a lengthy and expensive process with a high risk of human error. In these cases, we can resort to data-driven approaches.

Data-driven approaches are techniques that allow to automatically build a model by learning it from some data coming from the system. For example, these data may be in the form of a log of events, or a sequence of non-nominal values assumed by components. A data-driven approach to anomaly detection in an electrical power system is presented in [6]. The work aims at learning, using an autoassociative neural network, the nominal behaviour of the low level subcomponents of the system, and then use this model to identify anomalies that occur at any level of the system. Anomaly detection has also been applied to detect vulnerabilities and security threats in power system control centers. Instead of relying on intrusion detection systems, in [5] the nominal data flow and control operations of the system are modelled with a rough sets classification algorithm. Variations in the standard behaviour of the system are captured with the model and are an indication of a possible security threat. Anomaly detection has been applied in security context also in [7], where a graph-matching approach is applied to detect cyber attacks to smart grids. In [21] two machine learning methods were used to detect abnormal behaviours in a network of webcams and in a set of IP traffic statistics.

Most of today's information systems use a system log to record system states and significant events, which provides detailed information about the history of processes and can help to debug failures. However, because of the high number of entries produced by some systems, it can be extremely difficult to extract consistent information about the points of failure and the general behaviour of the system. This has prompted various studies aimed to identify anomalies by examining the data log. Deeplog [8] offers a data-driven approach to identify online attacks that views log entries as elements of a sequence which follow a specific grammar rules. A deep neural network automatically learns a model of the log patterns from the normal execution and is then able to report as anomalies any deviations from the standard system execution.

Process mining [22, 23] is an approach to event logs that focuses on discover, analyze, and improve business processes. These techniques are based on the assumption that each event refers to a specific activity (i.e., a well-defined step in some process) and belongs to a process instance. Additional informations such as the timestamp of the event of the device initiating the activity can be used by the technique. In general, there are three types of process mining analysis that can be executed: *discovery*, *conformance*, *enhancement*. *Discovery* produces a model from the log, without access to any prior knowledge on the system. *Conformance* allows to check if an already existing process model represents correctly the reality as shown in the log. *Enhancement* allows to improve an existing model using the information obtained from the event log. Process mining can also be adapted to perform anomaly detection. In [24] it is shown how the process mining framework ProM [25] can detect anomalous process traces.

The approaches to anomaly detection described in the literature and presented in this section cannot be directly applied to our problem. We have to exclude the application of model-based approaches since the electrical infrastructure we analyze is really complex and it would be a very difficult task to create a complete model for its nominal behaviour. Moreover, there is no human expert who knows everything about the electrical system, because it was built incrementally over many years. So, the objective of this thesis is to analyze failures and their consequences in terms of correlations between the electrical components. We cannot also apply directly most of the data-driven approaches presented, because they focus on building a model of the nominal behaviour of the system, although directly from data and without the contribution of a human expert. Most of the data entries of the log at our disposal are instances of irregularities of the system, with different levels of priority. Therefore we cannot build a nominal model from data since we do not have data that follows a standard behaviour. Instead, we model correlations about different faults. We will adopt Bayesian networks and Markov chains as tools to discover these correlations.

2.2 Bayesian networks

2.2.1 Model

Bayesian Networks are graphical probabilistic models that allow to graphically represent the dependence and independence probabilistic relationships between some variables. They are generally used in contexts in which it is necessary to model a reality that presents some degree of uncertainty, that is captured by the model with probabilities. Graphically, a Bayesian Network is a directed acyclic graph (DAG), in which each node represents a variable of the system and the edges between variables represents the direct influence among them. Figure 2.1 shows a simple Bayesian Network that correlates symptoms, illnesses and life style. For example, the variables "Exercise" and "Diet" directly influence the chance of "Heart Disease", while the "Heartburn" depends only by the "Diet".

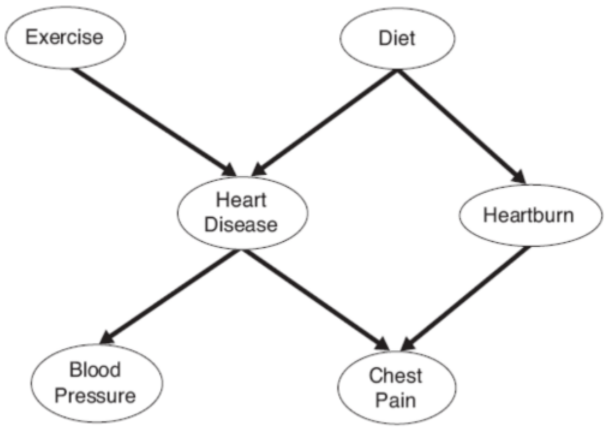


Figure 2.1: A Bayesian network taken from [1].

Each of the variables in Figure 2.1 has a domain that indicates the values they can assume. For example, the variable "Exercise" has $\{Yes, No\}$ as domain, while "Blood Pressure" can assume the values $\{High, Low\}$. Each variable in the model is also associated with a *conditional probability distribution* (CPD). Formally, the CPD of a variable Y given the variable X is the probability distribution of Y when X is observed to be a specific value. As shown in Figure 2.2, the CPD for a node X specifies the probability distribution over the values of X given all the possible assignment of values to its parents. The complete CPD tables are not shown in the picture, but every missing probability can be easily computed from the others.

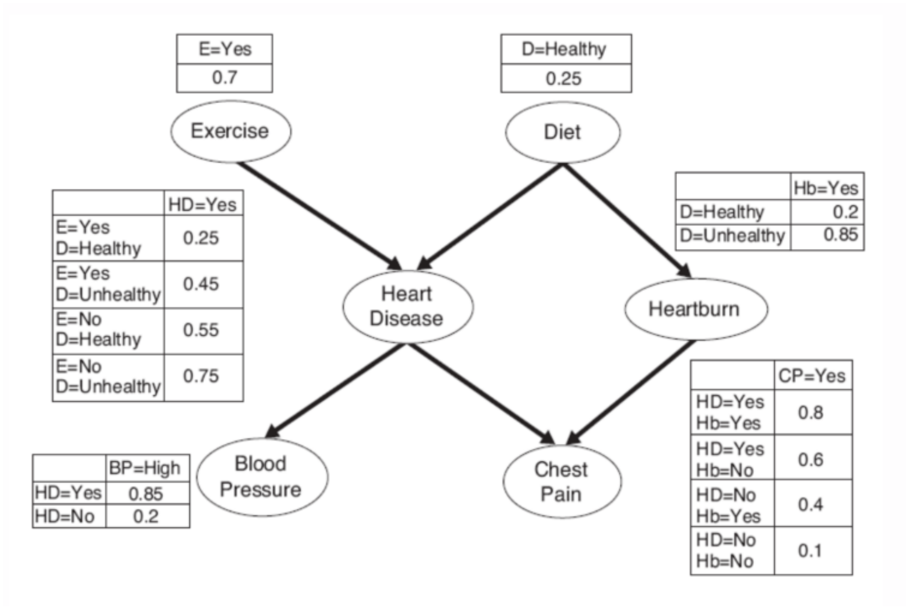


Figure 2.2: The Bayesian network shown in Figure 2.1 with the partial CPD attached to the nodes [1].

If a variable has no parents, the associated CPD turns into a marginal distribution. For example, the probability of the "Diet" being "Healthy" is 0.25, while the probability of it being "Unhealthy" is 0.75. These probabilities are expressed with the notation:

$$P(\mathit{Healthy}) = 0.25 \tag{2.1}$$

$$P(\mathit{Unhealthy}) = 0.75 \tag{2.2}$$

The conditional probability of the "Heart Disease" being "Yes" given the values of "Exercise" and "Diet" is expressed as:

$$\begin{split} P(\mathit{Heart\ Disease} = \mathit{Yes} \,|\, \mathit{Exercise} = \mathit{Yes}, \\ \mathit{Diet} = \mathit{Healthy}) = 0.25 \end{split} \tag{2.3}$$

Variables in Bayesian networks may have relation of *dependence* or *independence* with one to another. In general, a connection between two variables represents direct dependence, which means that variables always depend from their parents. However, observing the values of a variable may turn a dependence relation into a independence relationship: this is the concept of *conditional independence.* Referring to Figure 2.1, we can say that the "Blood Pressure" is conditionally independent from "Exercise", "Diet", "Heartburn" and "Chest Pain" given that we have observed the "Heart Disease":

$$\begin{gathered}(\textit{Blood Pressure} \perp \textit{Exercise}, \textit{Diet}, \textit{Heartburn}, \textit{Chest Pain} \\ \mid \textit{Heart Disease})\end{gathered} \tag{2.4}$$

Intuitively this reasoning makes sense, since the "Blood Pressure" depends directly only from the "Heart Disease", and once we have observed it any information from the other variables will not change our beliefs about the "Blood Pressure". This, however, does not mean that observing the values of a variable's parent makes that variable independent from all the others. For example, knowing the "Exercise" and "Diet" of a person does not make the "Heart Disease" independent from the "Blood Pressure", because information about the person's blood pressure can change our beliefs of how likely he is to have an heart disease. A more general pattern that emerges from these considerations is that *in a Bayesian network, each variable is independent from its non-descendants given that we have observed its parents* [2].

While dependencies and independencies can be precisely identified, the same can not be said about causality. An edge in the Bayesian network of the type $A \rightarrow B$ *can* imply causality (i.e., that A causes B), but it in general it simply means that there is a correlation between A and B. Representing causality is not easy, especially if the network structure has been learned automatically from the data. Moreover, even if we think that we have built a *causal Bayesian network*, there might be variables that we can't observe and we are not aware of, that can induce us to think a correlation as causal when the actual causal relation is with these hidden variables. Such variables are also called *latent variables* [2]. In general, a Bayesian network with a causal relationship among connected variables tends to have a sparser structure than pure "correlation" networks, but this does not influence the result of the standard probabilistic queries that we execute on our model. In fact, in this sense, it is not important if our model is causal or not, as long as the underlying distribution is represented correctly.

2.2.2 Inference

Answering queries to find out how probable is for some variables to assume certain values, possibly having observed the values assumed by other variables of the network (which are treated as *evidence*), is one of the main advantages and capabilities of Bayesian models.

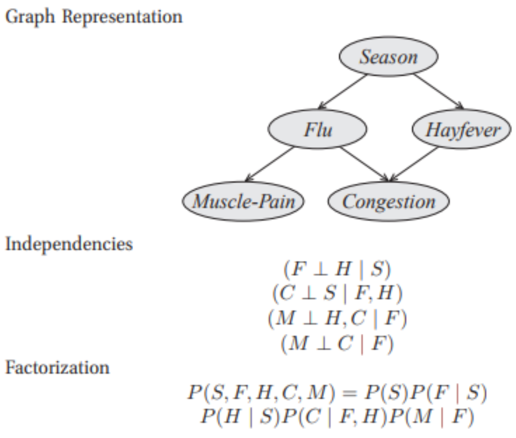


Figure 2.3: Another example of Bayesian Network related to a medical setting, taken from [2].

In Figure 2.3 we can see how a query related to all the variables in the domain can be factorized exploiting the known independencies among the variables, following the edges from the root to the leaves. The factorization in Figure 2.3 is a scheme to be adopted regardless of the values associated to each variable. A value to each variable has to be obviously associated to perform a query and the following *joint query* shows a possible instantiation of the scheme in the previous figure.

$$\begin{aligned} P(&Season = Winter, Flu = True, Hayfever = False,\\ &Congestion = False, Muscle\ Pain = False) \end{aligned} \tag{2.5}$$

A joint query calculates the joint probability of multiple values. The query in Equation (2.5) returns the probability of the 5 events joined in a logic "AND". A joint query can be rewritten as a product of *conditional probability queries*, as shown before in Figure 2.3. This type of query is useful to discover the probability of an event knowing that another event has occurred, and it's written in the following form:

$$P(X = x \mid E = e) \tag{2.6}$$

X is a generic variable that can assume the value x and $E = e$ is a set of assignment of values e to variables E (i.e., the evidence, or the observed variables). Without the factorization mechanism provided by Bayesian networks it would be hard to ask queries to discover information from domains composed of many variables, which can be easily encountered in a real world setting. Exploiting independencies can greatly reduce the probability space, which corresponds to the possible assignments to the variables of the domain. In this way the number of nonredundant parameters needed to define high dimensional joint distributions can be greatly reduced. Rather than encode the probability of every possible assignment in our domain, we can break up the distribution into smaller factors, each of them over a much smaller space of possibilities, and then define the joint distribution as a product of these factors [2].

A different kind of query is the *Maximum a Posteriori* (MAP) query:

$$MAP(X \mid E = e) = \underset{x}{\operatorname{argmax}}\, P(X{=}x \mid E{=}e) \tag{2.7}$$

The MAP query returns the most likely assignment of values x to the variables X. Of course, we can answer some of these queries directly by looking at the CPD, but in most of the cases this is not the most practicable or convenient approach. Another possible type of queries are the *intervention queries*, which can tell us what happens to a certain variable when we manipulate the values of other variables. For example, given some evidence, we can ask how would have changed the probability of seeing a specific variable-value assignment if we had observed another variable. These kind of queries are strongly related to the causal meaning of edges and therefore make sense only in causal Bayesian networks.

The act of answering queries is called *inference.* There are two approaches to inference: *exact inference* and *approximate inference.* In exact inference, we analytically compute the conditional probability distribution over the variables and we obtain the exact probabilities. Unfortunately, exact inference is a NP-hard problem and therefore there is no algorithm that can perform it efficiently in all the network configurations. For a faster, but less precise solution, we must resort to approximate inference, which is still an NP-hard problem in the worst case. However, many real-world applications can be tackled very effectively using exact or approximate inference [2]. One of the most popular exact inference method is the *variable elimination algorithm*, which computes the result of a query in the most efficient way, starting from the factorized representation and distributing sums over products. If we define a set X of variables of the domain, and a subset of variables $Y \subset X$ involved in a query, the factors related to the variables in the complementary subset $X - Y$ have to be summed, i.e., marginalized out of the distribution. The choice of the elimination ordering of the variables in order to minimize the computational time is one of key points in exact inference algorithms and its complexity is NP-hard, and can be tackled by using greedy algorithms. Dynamic programming can also be used to compute the innermost summations first, to avoid redundant computations [9]. Another method to perform exact inference is the *clique tree algorithm* that uses message passing in a preconstructed clique tree (or junction tree), a data structure for exact inference. Messages are passed between the cliques in the tree until all cliques agree on the same marginal beliefs of any variable they share [2].

Most of the algorithms that perform approximate inference adopt either a *sampling* or *message passing* approach. *Sampling* consists in approximating the probability of events by drawing samples from sampling distributions defined over the Bayesian network (conceptually similar to performing simulations). The value of the probabilities are then directly based on the occurrences of the events in the simulations. Sampling algorithms may differ in the way they generate samples and weight them to compute the event probabilities. Some examples of these techniques are *importance sampling* and *Markov chain Monte Carlo* methods [2]. On the other hand, *message passing* algorithms like *belief propagation* are based on the idea of passing messages along the nodes of the network until a stable belief state is reached, which corresponds to probability values that often provide a good approximation of the inference query results. This algorithm differs from the clique tree algorithm described for the exact inference because it relaxes some constraints for the construction of the tree, which in this case is a network called "cluster graph". For this reason the results obtained from this process are approximated [2].

2.2.3 Learning

Bayesian networks can be built manually, usually with the help of an expert, or automatically, by learning only the parameters or both the structure and the parameters directly from the data. Two of the main approaches to learn the structure of the network are *constraint based* algorithms and *search-and-score based* algorithms.

Constraint based methods look for the structure of the model that better explains all the conditional dependencies and independencies identified in the data with a statistical test. The disadvantage of these methods is that they can be sensitive to failures in individual independence tests. It suffices that one of these tests returns a wrong answer to mislead the network construction procedure. As a consequence, they are also less reliable when the number of samples is small.

On the contrary, *search-and-score based* algorithms utilize an heuristic based on a score function to search in the hypothesis space of all potential models and find the one that better fits the given data. The score function is what measures the fit to the data. Since this space is super-exponential in the number of variables, the problem is NP-hard and exact searches are infeasible in most instances. Local search algorithms as hill climbing, tabu search, simulated annealing and many others are usually used for this purpose. Those methods start from a candidate solution, which can also be a completely disconnected DAG, and iteratively modify it by adding or deleting an edge or inverting its direction, trying to build graphs with an increasing score value that is given by a score function. One of the disadvantages of score based methods is that they may incur in local maxima. With respect to constraint based methods they consider the entire network structure at once and they are less sensitive to the individual failures problems. The choice of adding or not an edge is less constrained by variable dependencies and is more dependent from the global network score.
The scoring functions used by the search methods can be *Bayesian scoring function* or based on *information theory scoring function.* K2 score, Bd score and Bdeu score are some of the most used Bayesian scoring functions. Starting from a prior probability distribution on the possible networks, they compute the posterior probability distribution $p(G|D)$ conditioned to the available data D, where G is the graph. The best network is the one that maximizes such posterior probability. In this case the probabilities for each training instance are evaluated in incremental order, and they contribute both to the evaluation of the model and to the final model score. The probabilities calculated for the m-th instance during the learning phase are influenced by the parameters learned from the first $m-1$ instances using Bayesian estimation. The m-th instance is used as a test case [2], and it becomes possible to train the model using all the available data, without the need to split data for doing training and testing, which can be an important issue

if we have limited amounts of data to learn from.
Scoring functions based on information theory are based on the minimum description lenght (MDL) principle, which establishes an appropriate trade-off between complexity and precision. This idea can be translated in the following family of scoring functions dependent on $f(N)$, a non-negative penalization function [26]:

$$g(G : D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \left(\frac{N_{ijk}}{N_{ij}} \right) - C(G)f(N) \tag{2.8}$$

r_i is the number of states of the variable X_i; the number of possible configurations of the parent set $Pa_G(X_i)$ of X_i is q_i; w_{ij}, $j = 1, ...qi$, represents a configuration of $Pa_G(X_i)$; N_{ijk} is the number of instances in the data set D where the variable X_i takes the value x_{ik} and the set of variables $Pa_G(X_i)$ take the value w_{ij}; N_{ij} is the number of instances in the data set where the variables in $Pa_G(X_i)$ take their j-th configuration w_{ij}; N_{ij} can also be written as $\sum_{k=1}^{r_i} N_{ijk}$; the total number of instances in D is N. $C(G)$ is the network complexity. It is proportional to the description length of the network and it is defined as the number of free parameters of the factorized joint probability distribution [26]:

$$C(G) = \sum_{i=1}^{n} (r_i - 1)q_i \tag{2.9}$$

Equation (2.8) can generate some famous scoring functions like AIC score ($f(N) = 1$), BIC score ($f(N) = \frac{1}{2}$) and maximum likelihood score ($f(N) = 0$). AIC and BIC scores select the network structure that best fits the data, penalized by the number of parameters which are necessary to specify the joint distribution. The network complexity term $C(G)$ is what implements the tradeoff between complexity and precision. The maximum likelihood score, instead, is not useful for learning the structure of a Bayesian network. In this case, the network complexity term $C(G)$ becomes no more relevant and simpler networks are never preferred over more complex ones. So, the choice of maximum likelihood score for structure learning causes the problem of *overfitting*. On the contrary, both AIC/BIC scores and the scores of the family of Bayesian scoring functions lead to the creation of networks which do not overfit. An important feature of both Bayesian and AIC/BIC score functions is that they are consistent: with a limited amount of data, simple structures with fewer edges have the highest scores; as the number of samples grows, more complex structures are preferred and asymptotically; with an infinite amount of data, the true model is selected [2]. A way to interpret scoring functions based on information theory is that they attempt to minimize the conditional entropy of each variable given its parents. So, they allow to search for the parent set of each variable that gives as much information as possible about this variable, or which most restricts the distribution. The addition of a penalization term is necessary because the minimum conditional entropy is always obtained after adding all the possible variables to the parent set [26].

Once we have the Bayesian network model, we can use other learning techniques to learn the value of the parameters (i.e., the conditional probabilities of the variables). Usually the aim is to find the *Maximum Likelihood Estimation* (MLE) of the parameters, which are the values that maximize the likelihood of the data.

As in the case of inference, the learning task becomes more difficult as the number of variables increases. Even the number of values that each variable can take contributes to the increased complexity. Clearly, continuous variables are harder to deal with than discrete variables.

2.2.4 Variations to the Bayesian model

Several models have been proposed over the years to extend and adapt Bayesian networks to different contexts with respect to the ones they were typically used in. For example, *dynamic Bayesian networks* are able to deal with the concept of time. In particular, they represent the dependencies of the variables from *previous* values of the variables by using *time slices*, where each time slice is conditionally dependent from the previous one. A dynamic Bayesian network is often called 2-TBN because it is a temporal Bayesian network (TBN) composed by two time slices. An example is shown in Figure 2.4 taken from [2]. Another variation of the Bayesian model is the *probabilistic relational model* (PRM). This representation extends the Bayesian networks by introducing classes, attributes, objects, and relationships among them. PRMs are able to model much more rich realities than standard Bayesian networks, but are also much more complex from a computational point of view, especially if the number of variables is high [2].

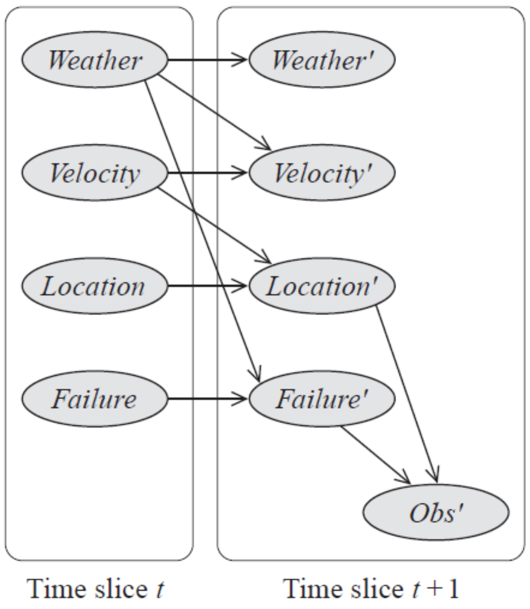


Figure 2.4: A 2-TBN taken from [2].

2.2.5 Applications

Bayesian networks have been widely used in practical applications. For several years, a Bayesian network was at the core of the VISTA system, a decision-theoretic system used in NASA mission control, predicting the probability of failures and advising the actions to take based on live data [9]. In [27] Kennet shows Bayesian networks case studies that range from biotechnology to web services and customer satisfaction surveys. [28] presents an overview of Bayesian networks applications in dependability, risk analysis, maintenance contexts and shows how both studies and practical applications of this methods are growing over time. [29] shows Bayesian networks applications in agriculture and suggests future developments of the techniques in that field.
A general scheme to utilize Bayesian networks in failure diagnosis has been presented by [30], while an expansion of the Bayesian model called "failure prediction Bayesian networks model" is discussed in [10] with the specific purpose of predicting failures. In [31] a Dynamic Bayesian network is used to model the causal relationships of faults in a petrochemical system; [32] shows how to build a causal Bayesian network that can be used in safety analysis.

2.3 Markov chains

A Markov chain is a stochastic model that represents the evolution of a process composed of a set of states $S = \{s_0, s_1, s_2, ...\}$. The process starts in one of these states and proceeds with a sequence of *steps* or *transitions*, where each *step* consists in transitioning from the current state to a new state, which could also coincide with the current state. The probability of this transition is called *transition probability* and is defined as $p_{s_i s_j}$, where $s_i, s_j \in S$. The dependence of the transitions on the current state only - and not on any previous state - is called *Markov property*. In order for the Markov chain to be valid, the transition probabilities must be such that:

$$\sum_{s_j}(p_{s_i s_j}) = 1 \qquad \forall i \in S \tag{2.10}$$

The transition probabilities can be compactly represented with a *transition matrix*. Graphically, a Markov chain is a state diagram where every edge between states is a transition. In Figure 2.5 we show an example of Markov chain and in Figure 2.6 the corresponding transition matrix.

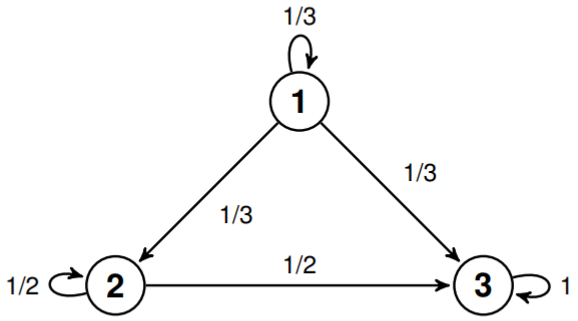


Figure 2.5: A simple example of Markov chain with 3 states, taken from [3].

$$\mathrm{T} = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 2.6: The transition matrix of the Markov chain in Figure 2.5.

Besides being able to show the evolution of a system and predict the probability of an immediate transition, Markov chains also allow to predict the probability of a sequence *seq* of states. To do this, we just have to compute the following:

$$p(seq) = \prod_{i=1}^{n-1} p(s_i|s_{i-1}) \tag{2.11}$$

Where i is the index of the state in the given sequence, n is the length of the sequence (that starts from index $i = 0$) and $p(s_i|s_{i-1})$ is the conditional probability of s_i given s_{i-1}, which corresponds to the transition probability $p_{s_{i-1}s_i}$. For example, in reference to Figure 2.5, the probability of seeing the sequence $seq = [1,2,2,3,3]$ is:

$$p(seq) = \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2} \cdot 1 = 0.075 \tag{2.12}$$

When we do not know the transition probabilities, we can learn them automatically, provided that we have data in the form of sequences of states. This is a simple task, since to find p_{ij} we just have to count the number of times that from the state i we observed a transition to state j, and divide this value by the number of total transitions that had i as starting point.

Despite being a quite simple model with respect to other graphical methods, Markov chains have found several interesting applications. One of the first applications is shown as an example by Markov himself [33] and consists in training the chain's transition probabilities on sequences of letters that compose a text, allowing to find the probability of vowel after a consonant, the probability of a consonant after a vowel, and so on. This application is also at the roots of automatic names generators that can be found in web sites and online multiplayer games. Markov chains are also part of Google's PageRank algorithm [34], which views web pages as states and links as transitions between them. In [11], Markov chains are used to perform fault detection by using a distributed group of agents whose individual transition matrices are exchanged and weighted based on other agent transition matrices. This exchange is used to reach a consensus about the state of the system and detect faults. In [12] Markov chains have also been used to model the nominal behaviour of an event log and detect anomalous log entries as deviations from the standard model. A similar approach was taken in [35], where Markov chains were used to detect intrusion in computer systems by learning the sequences of actions that normally are performed on the system, and using them do detect the irregular actions that are performed when an intrusion is in progress.

Chapter 3

Problem setting

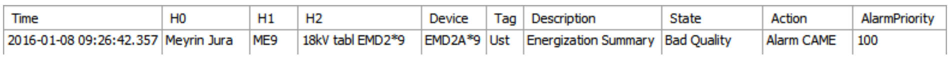
In this chapter we formalize the aspects of the problem we have worked on. In Section 3.1 we illustrate the main components of the system, in Section 3.2 we describe the format of the event log, and in Section 3.3 we formalize the problem setting. In Section 3.4 we describe more in detail the scope of this work and justify the choice of the techniques we have decided to utilize. In Section 3.5 we introduce the libraries chosen to learn and manipulate the models presented in this document.

3.1 The technical infrastructure

Our work is focused on a system composed of a large number of components that are called *devices*, each associated to a specific subsystem and location. During their operation, devices can raise alarms and diagnostic messages that are recorded in databases and that can be seen as events in a log. In this document, we will call *device* any kind of component that have reported an event in the log. Since the electrical system analyzed in this work has grown in complexity over time, with new components and areas progressively added and integrated within the existing infrastructure, it has become increasingly hard to keep track of specific correlations between devices in a global coherent model. A failure of a device, for example, could propagate and cause failures of other components or affect an entire subsystem. While a human expert could manually identify some of the affected devices, a tool able to automatically pinpoint the devices that *lead to* a failure and the devices that are *affected* by a failure could be extremely helpful to support the human experts, improve the overall understanding of the system, and facilitates its management.

3.2 The event log

We now show the structure of the event log. Each entry of the log is an *event* that is bound to a specific device of the system. Fields entries are filled automatically by the supervisory and monitoring system. Figure 3.1 shows a single entry of the log with the portion of the headers that are relevant to our analysis.



Time	H0	H1	H2	Device	Tag	Description	State	Action	AlarmPriority
2016-01-08 09:26:42.357	Meyrin Jura	ME9	18kV tabl EMD2*9	EMD2A*9	Ust	Energization Summary	Bad Quality	Alarm CAME	100

Figure 3.1: The fields of the event log with a single entry.

- **Time**: It indicates the timestamp at which the event occurred.
- **H0**: It denotes the geographical area where the device that generated this entry is situated.
- **H1**: It defines the system contained in H0.
- **H2**: It defines the subsystem involved.
- **Device**: It's a unique identifier for the device that has reported the alarm. We refer to this column when we talk about the name of the device.
- **Tag, Description, State**: These fields provide additional information on the event that was reported, for example on the state of the device that generated the event.

- **Action**: It determines if the alarm is becoming active or is terminated.
- **AlarmPriority**: The level of priority of this event: ranges from 0 to 130. The higher this value, the more relevant this event.

The "time" and the "device" identifier are the fields that are essential to our models. In all the queries done in our work, the entries will always be in ascending chronological order with respect to the timestamp in order to take into account temporal correlations among events. "Action" will be used only to select the events that are becoming active. The numerical value of "alarmPriority" has been discretized into a categorical variable **level of priority**, which takes four possible values: L0, L1, L2 and L3. L0 events are the ones with the lowest "AlarmPriority" value and range from 0 to 20, L1 events go from 21 to 59, L2 from 60 to 99 and L3 events are the most critical ones and range from 100 to 130. Apart from "time", "device" and "action", all the other fields can be integrated in different parts of the models and allow to deepen the analysis on a specific aspect of the correlation between devices.

3.3 Formal setting

Each one of the models that we will build is related to a single set of timestamps $T = \{t_1, t_2, t_3, ..., t_n\}$ that correspond to some *relevant situations* that have happened in the system. These relevant situations could be system failures, alarms risen by specific devices, or simply time instants that an operator believes to be interesting. For most of our work we will obtain the set T by extracting the timestamps of events of specific devices d_{ref}, that we call *reference devices*. The choice of the reference device and of its timestamps will be discussed in detail in the next chapter. In general, a reference device d_{ref} reports in the log a series of events $E_{ref} = \{e_1, e_2, e_3, ..., e_n\}$. Since each event $e_i \in E_{ref}$ is bound to an unique timestamp, we can easily find the set of timestamps T at which events in E_{ref} occur.

In relation to the event log structure illustrated in Section 3.2, we define a function for each of the log's fields, with the same name of such fields, that, given an event, returns the content of a field of that event. Given E as the set of all events in the log, in Equations (3.1) - (3.4) some of these functions are shown.

$$tag\colon E \rightarrow String \tag{3.1}$$

$$timestamp\colon E \rightarrow Date \tag{3.2}$$

$$state\colon E \rightarrow String \tag{3.3}$$

$$alarmPriority\colon E \rightarrow \mathbb{N} \tag{3.4}$$

We call these functions *field functions* and we say they belong to a set F of field functions. For example, the function $tag(e)$ applied to the event e shown in Figure 3.1 returns the string "Ust". Additionally, we introduce a function $events(d)$ that, given a device d, returns all the events of that device:

$$events(d) = \{e \in E \,|\, device(e) = d\} \tag{3.5}$$

We now consider a single timestamp $i \in T$, in the specific case in which we want to build a model that correlates alarms happening *after* the anomalies. We will reference Figure 3.2 for the following discussion. Since the timestamp i is a specific instant that can be identified in the

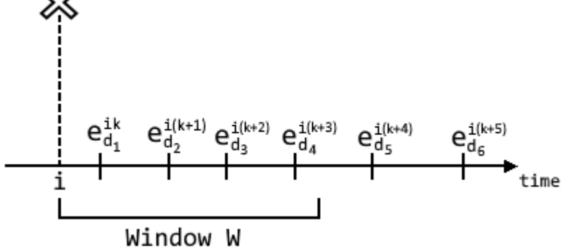


Figure 3.2: The timeline of some events happening after a timestamp i.

timeline of events reported in the log, it will generally be followed by events that can be related to any device d in the system. Let's define these events as $e^{ik}_{d_j}$, where d_j is the device that reported the event e, i indicates that this event is currently considered in relation with the timestamp i and k is a progressive integer number that uniquely identifies this event in the database. We now consider a time window W, called *analysis window*, that starts from the timestamp i and ends W time after. All the events $e^{ik}_{d_j}$ that lie inside this window are part of the set E^i_c, that is the set of events correlated to timestamp i, as shown in Equation (3.6).

$$\begin{aligned} E^i_c = \big\{e \in E \mid \mathit{timestamp}(e) \geq i, \\ \mathit{timestamp}(e) \leq i + W\big\} \end{aligned} \tag{3.6}$$

In the example, $E^i_c = \{e^{ik}_{d_1}, e^{i(k+1)}_{d_2}, e^{i(k+2)}_{d_3}, e^{i(k+3)}_{d_4}\}$. If we repeat this procedure for each timestamp $i \in T$, we can build a set E_c called *correlation set*:

$$E_c = \bigcup_{i=1}^{n} E^i_c \tag{3.7}$$

All the devices d_j related to at least an event $e^{ik}_{d_j} \in E_c$ compose the set C that we call *candidate devices.*
The same reasoning and naming convention can be easily repeated for the case in which we want to analyze events that happened *before* the anomalies. The main difference is that we will now talk of a correlation window $-W$. The composition of the set E^i_c for this case is defined in Equation (3.8) and an example of the timeline is shown in Figure 3.3.

$$\begin{aligned} E^i_c = \big\{e \in E \,|\, \mathit{timestamp}(e) \leq i, \\ \mathit{timestamp}(e) \geq i - W\big\} \end{aligned} \tag{3.8}$$

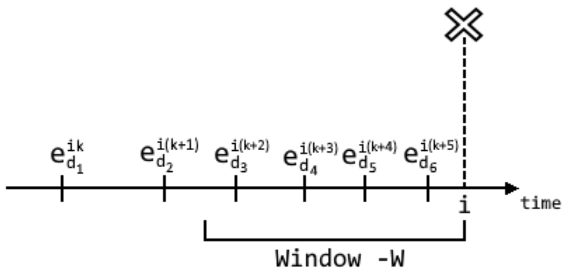


Figure 3.3: The timeline with events happening before a timestamp i.

3.4 Scope and techniques

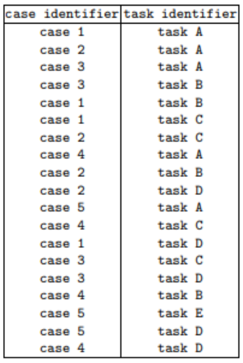
As explained in Section 2.1, differently from many other approaches to fault prediction and anomaly detection, in our work we do not build a nominal model of the system and then use it to identify anomalies and predict faults. Instead, we build models that represent what happens before or after specific events like alarms, failures, or simply instants of interest and we use it to predict correlated events that will rise in the near future or to detect previous events that caused the anomaly in the first place. More in detail, we generate two types of models over a subset of variables $V \subset C$ for each set of timestamps T at our disposal. We will see how these models complete each other and allow to obtain interesting information about the correlation between devices.

The first type of models chosen for our work is Bayesian network. Probabilistic graphical models like Bayesian networks offer the possibility to learn a model, to perform inference on it, and to represent it graphically, and there are few frameworks which offer these three opportunities together. Despite more complex models like the ones described in Section 2.2.4 exist, we believe that a standard Bayesian network is more fit in this context. For example, because of the availability of temporal data, one could think that using a Dynamic Bayesian network could be a good approach. However, to apply a DBN in our setting, we would need edges that link variables not only between consecutive time slices but also among non consecutive ones. This would make the model extremely complex, both in the number of variables and in the number of edges.

The second type of models chosen is Markov chain. The reason behind this choice is mainly because of its capability to describe sequentiality of events. In fact, as we will see in the next chapter, Bayesian networks can capture correlations and allow for complex inference but lack the capability of defining a temporal *order* of such correlations.

The fact that we are dealing with an event log provides both advantages and disadvantages. First of all, since every event ever happened is reported, there is a high amount of information available. Hopefully, every cause of every failure and all correlations between devices are present in the data. However, the obvious consequence is that having *too much* data makes information hard to extract. Correlations may indeed be present in the log, but they could be well hidden behind hundreds of non correlated events and noise. As described in Section 2.1, process mining is a technique specialized in processing data logs that is becoming more and more popular. However, it does not suits our problem very well. In fact, our log entries do not define the processes to which they belong; moreover, even if we could identify some general processes we would need to specify the precise steps that compose each process, which in many instances is impossible to do because of the non-deterministic nature of the electrical system's components. Ultimately, processes and their step are two minimal requirements of process mining that are not met in the context of our problem.
The table in Figure 3.4 taken from [36] is an example of workflow log which allows to perform process mining. Each event should refer to a case (process instance) and to a task, i.e., a well-defined step in the process. Our log unfortunately does not define any of these two required information.

The use of data mining techniques like association rules [37] and classification rules [38] has been also discussed and tried as a part of a



case identifier	task identifier
case 1	task A
case 2	task A
case 3	task A
case 3	task B
case 1	task B
case 1	task C
case 2	task C
case 4	task A
case 2	task B
case 2	task D
case 5	task A
case 4	task C
case 1	task D
case 3	task C
case 3	task D
case 4	task B
case 5	task E
case 5	task D
case 4	task D

Figure 3.4: Example of log needed to apply process mining.

previous, preliminary approach to the problem discussed in this work, but results of these techniques proved to be less successful than the compact, readable probabilistic model of the Bayesian networks. A similar motivation can be applied in favor of the use of Markov chains graphical representation of temporal sequences with respect to the list of frequent sequences obtained using sequential pattern mining [39].

3.5 Libraries

We have built Bayesian networks with the help of a Python library called "Pgmpy" which implements algorithms for exact inference (variable elimination, belief propagation), for performing probabilistic and MAP queries, and for both search-and-score based and constraint-based learning. The search strategies implemented are hill climbing and exhaustive search, while Bic, Bdeu, and K2 are the score metrics available [40]. To generate the Bayesian models shown in this document we have used hill climbing with the Bic scoring method and variable elimination to perform inference. For what concerns Markov chains we have used the "Pomegranate" Python library [41]. It allows to learn Markov chains from data and to calculate the probability of any sequence given in input by an user. The graph visualization of both Bayesian networks and Markov chains have been performed with the “Graphviz” library [42], while the well-known clustering algorithms that will be used in the next chapter have been implemented with scikit-learn [43].

Chapter 4

The Bayesian network model

In this chapter we show the steps to build a Bayesian network that can represent correlations between devices. In Section 4.1 we describe all the preliminary choices and passages needed to build the model. In Section 4.2 we generate the structure of the network and train its parameters, while in Section 4.3 we show some graphical alterations that make the Bayesian networks more meaningful in our context. The methods presented in the chapter are summarized in the Table 6.1 in Section 6.1.

4.1 Pre-processing of data

4.1.1 The reference devices

As we anticipated in Section 3.3, for most of our work we will extract the set T considering some devices that we call "reference devices". A reference device is a device in the electrical system that is interesting according to some criterion. For example, it could be a device that holds some importance for the infrastructure because of its critical role in the system, or it could be a device that frequently reports events in the log and an analysis is needed to understand its behaviour. A general technique that we will use to identify interesting devices without

resorting to an expert guide is to select devices that have the highest number of events within the analysis window W after or before each of their own events. We can also find the devices that are followed by the highest number of events with a specific "level of priority", which can be considered as components of critical importance if the priority level is high. It is obviously true that the sole quantity of reports after the events of a reference device could not be enough to denote that device as interesting, but behind this choice there is also a practical reason: in order to build an effective prediction model we need an high number of data to perform the training phase.
In general, when we select a reference device for our analysis we will always consider only its events with one of the four "levels of priority". This means that the set of timestamps T will be composed only by events with a certain priority. This choice is important because the consequence of an event with priority "L0" is, for example, usually unrelated to the consequence of an event of priority "L2".

4.1.1.1 Multiple reference devices

Sometimes we might be interested in analyzing the correlation between a group of devices D without having at disposal a set T of timestamps. In this case we can compose T by taking all the timestamps of all the events in D:

$$T = \bigcup_{e_d \in \text{events}(d), d \in D} \text{timestamp}(e_d) \tag{4.1}$$

Of course, to actually perform the analysis of correlations on these devices we need to select them as variables in our network ($V = D$).

4.1.2 The analysis window

An important parameter to choose is the length of the analysis window W. This value determines how many events we will save in the itemsets E_c^i and, as a consequence, which candidate devices C will be available to be used as variables in our models. Unless we are given a specific itemset T and some prior knowledge on the devices that could be correlated, the choice of W has to be made by taking an informed guess based on the data distribution. Preliminary analysis on the event log suggested that a value between 1 and 10 minutes could be appropriate. For the rest of this document we will assume $W = 5$ minutes, and every network shown is to be considered as generated after the timestamps, unless otherwise specified. We will discuss the validity of this decision in Chapter 6.

4.1.3 The training set generation

In order to build the training set that will be used to generate the structure and set the parameters of the Bayesian network, we must first define what is a variable in this model. In its simplest form, a variable in our Bayesian networks is a device and can assume binary values: 1 or 0. Among all the possible candidates in C, we have chosen a subset V that will become the variables of our model, i.e., the nodes of the Bayesian network. Details on how this choice is performed are shown in Section 4.2.1. For now, let's assume that we already have V. In Section 3.3 we showed how the sets E_c^i contain all the events related to a timestamp $i \in T$ within the analysis window W. Since each event $e_{d_j}^{ik} \in E_c^i$ is associated to one and only one device d_j, we can create the set $C_i = \{d_1, d_2, ..., d_m\}$ that contains all the devices that appear in events after or before a single timestamp i. Now we can define our training set S with the example in Equation (4.2):

$$\mathbf{S} = \begin{array}{c} \\ 1 \\ 2 \\ .. \\ i \\ .. \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} d_1 & d_2 & .. & d_k & .. & d_m \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ .. & .. & .. & .. & .. & .. \\ 0 & 0 & 0 & 0 & 1 & 1 \\ .. & .. & .. & .. & .. & .. \\ 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{4.2}$$

We see that each row of the training set S is a *training instance* related a specific timestamp $i \in T$. For simplicity, we used natural numbers instead of timestamps in the example. Each column of S, instead, is related to a single device $d_j \in V$. A cell (i, d_j) contains 1 if the device d_j was found in C_i and 0 otherwise. When it is not otherwise specified, all the methods presented refer to this technique to generate the itemset and all the networks shown in the following are trained with the above approach.

4.1.3.1 The duplicate problem

If we just apply the training set generation method previously described to all the timestamps in T, we may duplicate some of our data. Consider the example shown in Figure 4.1. Here, two times-

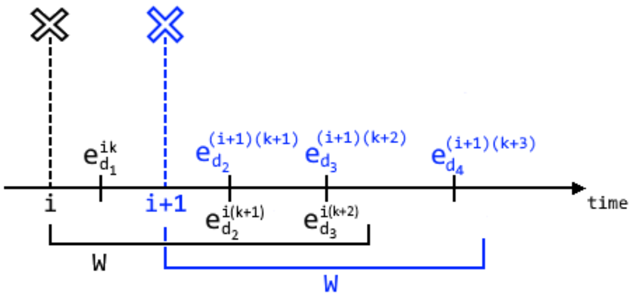


Figure 4.1: A fragment of the event log with the timestamp i (in black) and $i+1$ (in blue).

tamps $i, i+1 \in T$ are such that their temporal distance is less than W, which means that some of the events in the sets E_c^i and E_c^{i+1} are overlapping ($E_c^i \cap E_c^{i+1} \neq \{\}$). In particular, in the example we have:

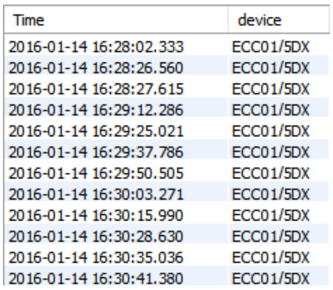
$$\begin{aligned} E_c^i &= \{e_{d_1}^{ik}, e_{d_2}^{i(k+1)}, e_{d_3}^{i(k+2)}\} \\ E_c^{i+1} &= \{e_{d_2}^{(i+1)(k+1)}, e_{d_3}^{(i+1)(k+2)}, e_{d_4}^{(i+1)(k+3)}\} \end{aligned} \tag{4.3}$$

but because of the equivalence:

$$\begin{aligned} e_{d_2}^{i(k+1)} &= e_{d_2}^{(i+1)(k+1)} \\ e_{d_3}^{i(k+2)} &= e_{d_3}^{(i+1)(k+2)} \end{aligned} \tag{4.4}$$

we have that the two sets have the events related to the devices d_2 and d_3 in common. This overlap happens quite frequently in the data we used in our experiments, sometimes with events duplicated up to dozens of times. A device affected by this behaviour is shown in Figure 4.2. The technical reason for this is actually not surprising. For example, when a device reports an anomaly, it might report other events shortly after, that differ in the content of other fields - like “tag” or “state” - which specify some other details about the first event or consequent anomalies. Figure 4.3 shows the same device of Figure 4.2 with the fields “tag” and “description”.

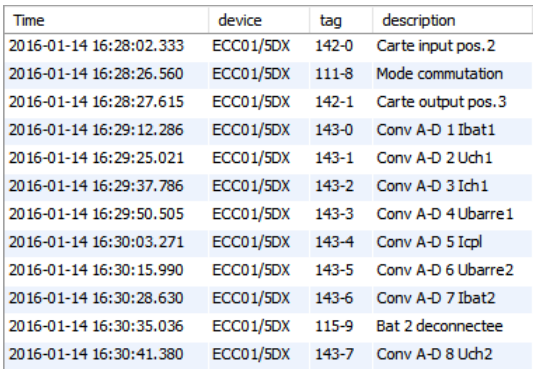
From the point of view of the training phase, we have that d_2 and



Time	device
2016-01-14 16:28:02.333	ECC01/5DX
2016-01-14 16:28:26.560	ECC01/5DX
2016-01-14 16:28:27.615	ECC01/5DX
2016-01-14 16:29:12.286	ECC01/5DX
2016-01-14 16:29:25.021	ECC01/5DX
2016-01-14 16:29:37.786	ECC01/5DX
2016-01-14 16:29:50.505	ECC01/5DX
2016-01-14 16:30:03.271	ECC01/5DX
2016-01-14 16:30:15.990	ECC01/5DX
2016-01-14 16:30:28.630	ECC01/5DX
2016-01-14 16:30:35.036	ECC01/5DX
2016-01-14 16:30:41.380	ECC01/5DX

Figure 4.2: A portion of the log with events of device ECC01/5DX extremely close one to the other.

d_3 are seen together two times instead of one, which increases the belief that they should be correlated. This results in Bayesian networks with CPD probabilities much biased than without duplicates. On the other hand, eliminating the duplicates from the timestamp $t+1$ may excessively weaken the correlations. In relation to Figure 4.1, consider the training set S_1 with duplicates in Equation (4.5) and S_2 without



Time	device	tag	description
2016-01-14 16:28:02.333	ECC01/5DX	142-0	Carte input pos.2
2016-01-14 16:28:26.560	ECC01/5DX	111-8	Mode commutation
2016-01-14 16:28:27.615	ECC01/5DX	142-1	Carte output pos.3
2016-01-14 16:29:12.286	ECC01/5DX	143-0	Conv A-D 1 Ibat1
2016-01-14 16:29:25.021	ECC01/5DX	143-1	Conv A-D 2 Uch1
2016-01-14 16:29:37.786	ECC01/5DX	143-2	Conv A-D 3 Ich1
2016-01-14 16:29:50.505	ECC01/5DX	143-3	Conv A-D 4 Ubarre1
2016-01-14 16:30:03.271	ECC01/5DX	143-4	Conv A-D 5 Icpl
2016-01-14 16:30:15.990	ECC01/5DX	143-5	Conv A-D 6 Ubarre2
2016-01-14 16:30:28.630	ECC01/5DX	143-6	Conv A-D 7 Ibat2
2016-01-14 16:30:35.036	ECC01/5DX	115-9	Bat 2 deconnectee
2016-01-14 16:30:41.380	ECC01/5DX	143-7	Conv A-D 8 Uch2

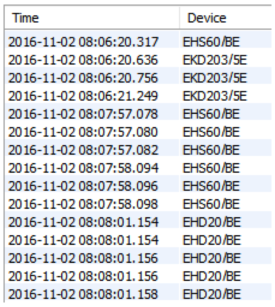
Figure 4.3: The log shown in Figure 4.2 with the addition of "tag" and "description".

duplicates in Equation (4.6). Suppose that $D = \{d_1, d_2, d_3, d_4\}$.

$$\mathbf{S_1} = \begin{array}{c c} & \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{array}{c} i \\ i+1 \end{array} & \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \tag{4.5}$$

$$\mathbf{S_2} = \begin{array}{c c} & \begin{array}{cccc} d_1 & d_2 & d_3 & d_4 \end{array} \\ \begin{array}{c} i \\ i+1 \end{array} & \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \tag{4.6}$$

The second row of matrix S_2 shows a training instance in which no correlation between d_4 and d_3 or d_4 and d_2 is present, even if in the timeline it appears that d_4 could be correlated with them.
Now consider the extract of the event log shown in Figure 4.4, whose events happen all within a 2 minutes interval. Of the three devices shown, consider EHS60/BE as reference device, so that its 7 events' timestamps are all in the set T. For simplicity, let's denote device EKD203/5E as d_{203} and device EHD20/BE as d_{20}. Without duplicates,



Time	Device
2016-11-02 08:06:20.317	EHS60/BE
2016-11-02 08:06:20.636	EKD203/5E
2016-11-02 08:06:20.756	EKD203/5E
2016-11-02 08:06:21.249	EKD203/5E
2016-11-02 08:07:57.078	EHS60/BE
2016-11-02 08:07:57.080	EHS60/BE
2016-11-02 08:07:57.082	EHS60/BE
2016-11-02 08:07:58.094	EHS60/BE
2016-11-02 08:07:58.096	EHS60/BE
2016-11-02 08:07:58.098	EHS60/BE
2016-11-02 08:08:01.154	EHD20/BE
2016-11-02 08:08:01.154	EHD20/BE
2016-11-02 08:08:01.156	EHD20/BE
2016-11-02 08:08:01.156	EHD20/BE
2016-11-02 08:08:01.158	EHD20/BE

Figure 4.4: A portion of the log with devices EHD20/BE, EHS60/BE, EKD203/5E.

we would obtain the following training set in Equation (4.7):

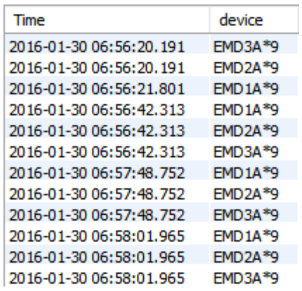
$$\mathbf{S_1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cc} d_{20} & d_{203} \end{array} \\ \begin{pmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \tag{4.7}$$

which shows only a correlation between d_{20} and d_{203}. In the other 6 timestamps d_{20} is always 0 because it has been already considered after the first timestamp. The training set with duplicates, instead, is in Equation (4.8):

$$\mathbf{S_1} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{cc} d_{20} & d_{203} \end{array} \\ \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 0 \end{pmatrix} \end{array} \tag{4.8}$$

Now d_{20} appears alone 6 times, with no correlations with d_{203}. The immediate consequence is that in this example the introduction of duplicates weakens the relationship between the two devices. In fact, without duplicates, we see that when d_{20} reports an event, d_{203} always reports an event. With the duplicates, when d_{20} reports an event, in most of the cases (6 over 7) d_{203} does not report any event.
The conclusion of all the previous considerations is that by ignoring the duplicates we can still find all the correlations with the possible downside of making some of them stronger and some of them weaker; by eliminating the duplicates we may weaken some correlations with the advantage of not over-representing any of them. These two approaches are both valid, and can lead to slightly different networks. In the rest of the document, we will always specify when we are using one solution or the other.

Sometimes, however, neither of the two approaches above can tackle the duplicate problem. Consider the devices shown in Figure 4.5. After a quick inspection we can easily see that the events of these three devices seem to always appear together, usually even in the very same timestamp. Regardless of how trivial this correlation might be (as the names of the devices suggests), we can notice that the four groups of triplets are all happening in an interval of less than 5 minutes (the length of our window of analysis). If one of the timestamps $i \in T$ was to be right before the first event shown in the log, we would obtain



Time	device
2016-01-30 06:56:20.191	EMD3A*9
2016-01-30 06:56:20.191	EMD2A*9
2016-01-30 06:56:21.801	EMD1A*9
2016-01-30 06:56:42.313	EMD1A*9
2016-01-30 06:56:42.313	EMD2A*9
2016-01-30 06:56:42.313	EMD3A*9
2016-01-30 06:57:48.752	EMD1A*9
2016-01-30 06:57:48.752	EMD2A*9
2016-01-30 06:57:48.752	EMD3A*9
2016-01-30 06:58:01.965	EMD1A*9
2016-01-30 06:58:01.965	EMD2A*9
2016-01-30 06:58:01.965	EMD3A*9

*Figure 4.5: A portion of the log with 3 devices: EMD1A*9, EMD2A*9 and EMD3A*9.*

a single training instance with all 3 devices set to 1, independently from the duplicate approach taken. But the log suggests that this correlation happened 4 times, so we now believe that there should be 4 training instances related to the timestamp i, instead of only one. Unfortunately, there is no way to smartly solve this problem with the current training set generation technique, unless we resort to another approach: clustering.

4.1.3.2 Clustering

Clustering can be seen as both a way to reduce the duplicate problem explained in the previous paragraph as well as a different way in which we see correlations between devices. Instead of assuming all events happened within 5 minutes as correlated, we now suppose that correlated events *appear in groups*, and in the same window of analysis we may find multiple groups such that a device in each group is correlated to devices appearing in the same group but not with devices appearing in other groups. This approach is also based on an important assumption: that the correlation between non consecutive events in the same cluster does not depend by how temporally far away are such events, but only by their belonging to the same cluster. This means that with clustering methods we could put in the same itemset two events e_1 and e_2 that are 1 minute apart from each other, provided that there's a high number of events between them; and at the same time put in different clusters events e_2 and e_3 that are only 10 seconds one from the other. It might seem incorrect at first, but it could be fine if we consider the electrical infrastructure that lies beneath the generation of these events. Particular states of devices, possibly caused by external events, can cause a sequence of alarms and anomalies all very close to each other, even a few milliseconds apart. When the sequence is over, another one may trigger, possibly for different reasons. Obviously there's the possibility that the new sequence of frequent events is actually related to the previous one, or to a specific device in such sequence, but we can assume that devices in such new sequence have an higher chance to be correlated among them than to be correlated with devices in the previous sequence. Training set generation with clustering can therefore produce quite different results than the one defined in Section 4.1.3, and might identify a whole new set of correlations.

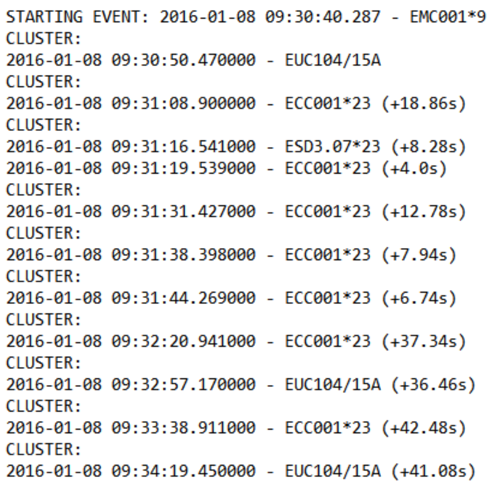
In order to adopt the clustering approach we directly apply clustering techniques to the events E_i^c for all i. Until now the number of rows of the training set S coincides with the cardinality of T, because all the events inside the window W starting from a timestamp i are saved in the same itemset C_i. We can redefine the generation of the itemsets by dividing every set of events E_c^i in smaller subgroups E_c^{il}, where l identifies a cluster and takes values from 1 up to q_i, which is the number of clusters found in the event set E_c^i. The value of q_i cannot be known in advance and will be provided directly by the clustering technique used.
We now define Q as the total number of clusters across all timestamps:

$$Q = \sum_{i \in T} q_i \tag{4.9}$$

Similarly to what we have done for the itemsets generation in Section 4.1.3, since each event $e_{d_j}^{ik} \in E_l^c$ is associated to only one device d_j, we can build a set of devices $C_{il} = \{d_1, d_2, ..., d_m\}$ that contains all the devices related to the timestamp i that appear in a cluster l. While the number of columns of the training set S remains m, the number of rows will be Q, which is greater or equal than n. Each row of the training set S is not anymore related to a specific timestamp $i \in T$, but to a specific cluster. A cell of S, (il, d_j), contains 1 if the device d_j is found in C_{il} and 0 otherwise.

Now that we have defined the way in which the training set is composed, we must decide how the actual clustering will be performed. Generally, we consider in the same cluster the events that happen consecutively one after another in a time interval that is considerably smaller than time intervals between non consecutive devices. To reach this objective we have tested different clustering algorithms: from simple techniques based on average and standard deviations of temporal distances between consecutive events to well-known clustering algorithms like *DBSCAN* and *Mean Shift*.

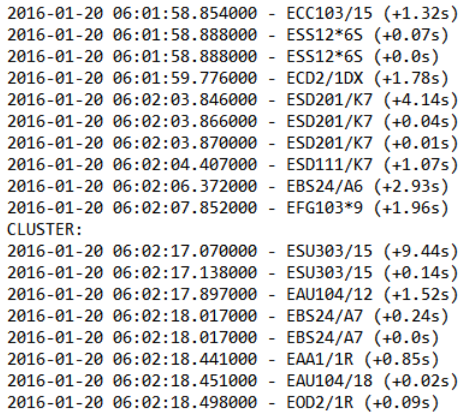
Fixed distance The simplest method consists in separating events in the E_c^i sets in such a way that the temporal distance between the last event of the cluster l and the first event of the cluster $l + 1$ is bigger than a predefined value, for example 5 seconds. Inside a cluster, every event is not more temporally distant from its predecessor and successor than the value chosen. We call this method *fixed distance clustering*. In Figure 4.6 and 4.7 we show some results obtained with this algorithm using 5 seconds as fixed distance *fd*. We can see in Figure 4.6 how the 5 seconds limit sometimes separates events that we might want to keep in the same cluster, like the ones happened at 9:31:38 and 9:31:44. In fact, the time interval between these two events is closer with respect to the time intervals between the other events in Figure 4.6, in particular the last four which are more distant than 30 seconds. For this reason, we are convinced that the events happened at 9:31:38 and 9:31:44 are more likely to be correlated with each other given the higher average temporal distance between the neighboring events. Following this assumption, we think that the last four events are not correlated to each other because their time distance is higher



```
STARTING EVENT: 2016-01-08 09:30:40.287 - EMC001*9
CLUSTER:
2016-01-08 09:30:50.470000 - EUC104/15A
CLUSTER:
2016-01-08 09:31:08.900000 - ECC001*23 (+18.86s)
CLUSTER:
2016-01-08 09:31:16.541000 - ESD3.07*23 (+8.28s)
2016-01-08 09:31:19.539000 - ECC001*23 (+4.0s)
CLUSTER:
2016-01-08 09:31:31.427000 - ECC001*23 (+12.78s)
CLUSTER:
2016-01-08 09:31:38.398000 - ECC001*23 (+7.94s)
CLUSTER:
2016-01-08 09:31:44.269000 - ECC001*23 (+6.74s)
CLUSTER:
2016-01-08 09:32:20.941000 - ECC001*23 (+37.34s)
CLUSTER:
2016-01-08 09:32:57.170000 - EUC104/15A (+36.46s)
CLUSTER:
2016-01-08 09:33:38.911000 - ECC001*23 (+42.48s)
CLUSTER:
2016-01-08 09:34:19.450000 - EUC104/15A (+41.08s)
```

*Figure 4.6: Fixed distance clustering applied to E_c^i events following timestamp i=2016-01-08 09:30:40.287 of the reference device EMC001*9*

than the events preceding them in Figure 4.6, and so we think that the fixed distance clustering has made the correct decision of separating these four events in four different clusters. We also see that there are many events related to device ECC001*23, which presents the duplicate behaviour described in Section 4.1.3.1. Creating many clusters that contain just a few devices has a strong impact on the learning phase of Bayesian model as well as the inference process, because the training set matrix S will be sparser, composed by many rows with few 1 and many 0. In the example, the standard training set generation of Section 4.1.3 would have produced a single itemset C_i (with $i = 2016 - 01 - 0809 : 30 : 40.287$) containing a 1 for each of the devices shown in the image (supposing that they all belong to V). With clustering we now have 10 training instances that almost all contain only one element, that therefore cannot properly model any correlation between devices. Figure 4.7 shows an example of an instance in which fixed distance clustering provides good results because most of the events happen only a few seconds one after the other. Of course it is possible



```
2016-01-20 06:01:58.854000 - ECC103/15 (+1.32s)
2016-01-20 06:01:58.888000 - ESS12*6S (+0.07s)
2016-01-20 06:01:58.888000 - ESS12*6S (+0.0s)
2016-01-20 06:01:59.776000 - ECD2/1DX (+1.78s)
2016-01-20 06:02:03.846000 - ESD201/K7 (+4.14s)
2016-01-20 06:02:03.866000 - ESD201/K7 (+0.04s)
2016-01-20 06:02:03.870000 - ESD201/K7 (+0.01s)
2016-01-20 06:02:04.407000 - ESD111/K7 (+1.07s)
2016-01-20 06:02:06.372000 - EBS24/A6 (+2.93s)
2016-01-20 06:02:07.852000 - EFG103*9 (+1.96s)
CLUSTER:
2016-01-20 06:02:17.070000 - ESU303/15 (+9.44s)
2016-01-20 06:02:17.138000 - ESU303/15 (+0.14s)
2016-01-20 06:02:17.897000 - EAU104/12 (+1.52s)
2016-01-20 06:02:18.017000 - EBS24/A7 (+0.24s)
2016-01-20 06:02:18.017000 - EBS24/A7 (+0.0s)
2016-01-20 06:02:18.441000 - EAA1/1R (+0.85s)
2016-01-20 06:02:18.451000 - EAU104/18 (+0.02s)
2016-01-20 06:02:18.498000 - EOD2/1R (+0.09s)
```

Figure 4.7: Other results from the application of fixed distance clustering with fd = 5 seconds.

that a more correct representation of correlations should have kept all the shown events in the same cluster, instead of separating them between timestamps $6:02:07$ and $6:02:17$. Or maybe, given the event density in a short time, the events that happen together in a timespan of less than 0.5 seconds should have been grouped separately and the right *fd* value could have been lower. If we don't know a precise temporal value under which two devices can influence each other, the fixed distance clustering method has to be discarded or improved by choosing a dynamic threshold for the separation, directly dependent from the particular data available.

Average and standard deviation Instead of choosing a fixed temporal value as a separation criterion, we could use some statistical measures like average and standard deviation. Given two events $e_1, e_2 \in E_c^i$ we now formally define the temporal distance between them d_{e_1,e_2}:

$$d_{e_1,e_2} = timestamp(e_2) - timestamp(e_1) \tag{4.10}$$

and the Boolean function $cons(e_1, e_2)$ that returns true if e_2 is immediately after e_1 in the event log. The set D_i of temporal distances between events related to a timestamp i is defined as:

$$D_i = \bigcup_{cons(e_1,e_2),e_1,e_2 \in E_c^i} d_{e_1,e_2} \qquad \forall i \in T \tag{4.11}$$

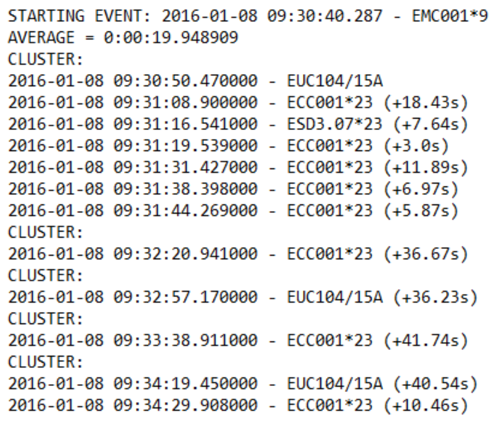
We can now compute the average $avg(E_c^i)$ of the entire set of events E_c^i:

$$avg(E_c^i) = \frac{\sum\limits_{d \in D_i} d}{|D_i|} \tag{4.12}$$

and the standard deviation $stdev(E_c^i)$:

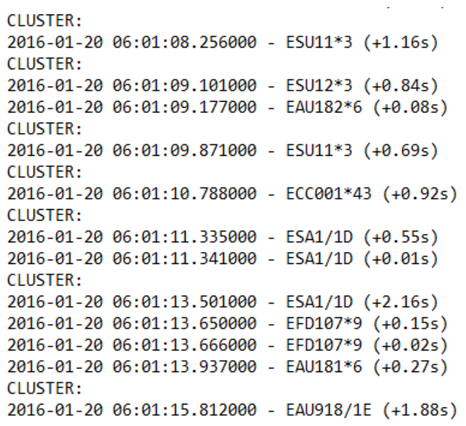
$$stdev(E_c^i) = \sqrt{\frac{\sum\limits_{d \in D_i} (d - avg(E_c^i))^2}{|D_i|}} \tag{4.13}$$

We now follow the same clustering procedure of the fixed distance clustering, with the only difference that we substitute the fixed distance *fd* with the function $avg(E_c^i)$. We call the new algorithm *average based clustering.* This new algorithm applied to the same data in Figure 4.6 provides much better results than fixed distance clustering, as shown in Figure 4.8. In Figure 4.6 the data was excessively divided into clusters because the average time interval between event occurrences was much higher than the value *fd* chosen, while with the average separation data are compacted when this is reasonable. Average based clustering, instead, does not perform well when the values in the set D_i tend to assume similar values, like in Figure 4.9, that shows only a significative part of the events in the set E_c^i clustered according to the average based technique, where i=2016-01-20 06:00:19.437. An high number of events related to that timestamp are registered, and the value of $avg(E_c^i)$ is 0.28 seconds, which is too low. A little change in the density of events inside the same set E_c^i can lead to an excessive separation of some of the events inside a huge set, a part of which has been shown in Figure 4.9. Average based clustering is still a great improvement with respect to the fixed distance clustering, but we need a more robust separation criterion. A solution could be the use of $k \cdot avg(E_c^i)$, with $k \in \mathbb{Q}^+$.



```
STARTING EVENT: 2016-01-08 09:30:40.287 - EMC001*9
AVERAGE = 0:00:19.948909
CLUSTER:
2016-01-08 09:30:50.470000 - EUC104/15A
2016-01-08 09:31:08.900000 - ECC001*23 (+18.43s)
2016-01-08 09:31:16.541000 - ESD3.07*23 (+7.64s)
2016-01-08 09:31:19.539000 - ECC001*23 (+3.0s)
2016-01-08 09:31:31.427000 - ECC001*23 (+11.89s)
2016-01-08 09:31:38.398000 - ECC001*23 (+6.97s)
2016-01-08 09:31:44.269000 - ECC001*23 (+5.87s)
CLUSTER:
2016-01-08 09:32:20.941000 - ECC001*23 (+36.67s)
CLUSTER:
2016-01-08 09:32:57.170000 - EUC104/15A (+36.23s)
CLUSTER:
2016-01-08 09:33:38.911000 - ECC001*23 (+41.74s)
CLUSTER:
2016-01-08 09:34:19.450000 - EUC104/15A (+40.54s)
2016-01-08 09:34:29.908000 - ECC001*23 (+10.46s)
```

*Figure 4.8: Average based clustering applied to events following timestamp i=2016-01-08 09:30:40.287 of the reference device EMC001*9*

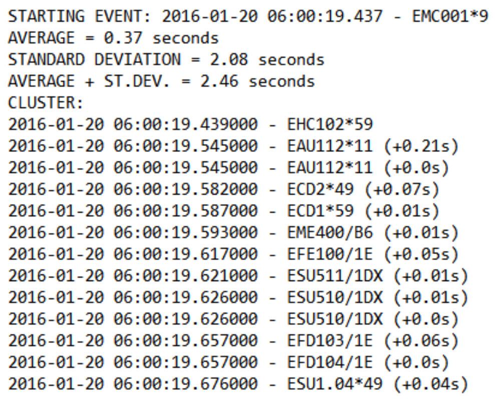


```
CLUSTER:
2016-01-20 06:01:08.256000 - ESU11*3 (+1.16s)
CLUSTER:
2016-01-20 06:01:09.101000 - ESU12*3 (+0.84s)
2016-01-20 06:01:09.177000 - EAU182*6 (+0.08s)
CLUSTER:
2016-01-20 06:01:09.871000 - ESU11*3 (+0.69s)
CLUSTER:
2016-01-20 06:01:10.788000 - ECC001*43 (+0.92s)
CLUSTER:
2016-01-20 06:01:11.335000 - ESA1/1D (+0.55s)
2016-01-20 06:01:11.341000 - ESA1/1D (+0.01s)
CLUSTER:
2016-01-20 06:01:13.501000 - ESA1/1D (+2.16s)
2016-01-20 06:01:13.650000 - EFD107*9 (+0.15s)
2016-01-20 06:01:13.666000 - EFD107*9 (+0.02s)
2016-01-20 06:01:13.937000 - EAU181*6 (+0.27s)
CLUSTER:
2016-01-20 06:01:15.812000 - EAU918/1E (+1.88s)
```

*Figure 4.9: An extract of the application of average based clustering to events following timestamp i=2016-01-20 06:00:19.437 of the reference device EMC001*9.*

However, this poses the problem of tuning the parameter k. For some $i \in T$, the choice of $k = 1$ could perform better in creating meaningful clusters than $k = 2$ and vice versa.

We can introduce instead an additional modification of the algorithm that takes $avg(E_c^i) + stdev(E_c^i)$ as separation criterion. The use of the $stdev(E_c^i)$ is important because the more the temporal distance values are variable, the more the algorithm groups together temporally distant events. On the contrary, if the temporal differences are regular, the $avg(E_c^i)$ term becomes more relevant. The criterion based on the sum $avg(E_c^i) + stdev(E_c^i)$ experimentally performs better than any fixed numerical value and is more stable and statistically meaningful than separation approaches based only on $k \cdot avg(E_c^i)$, with $k \in \mathbb{Q}^+$. Figure 4.10 shows a portion of a clustering procedure performed after



```
STARTING EVENT: 2016-01-20 06:00:19.437 - EMC001*9
AVERAGE = 0.37 seconds
STANDARD DEVIATION = 2.08 seconds
AVERAGE + ST.DEV. = 2.46 seconds
CLUSTER:
2016-01-20 06:00:19.439000 - EHC102*59
2016-01-20 06:00:19.545000 - EAU112*11 (+0.21s)
2016-01-20 06:00:19.545000 - EAU112*11 (+0.0s)
2016-01-20 06:00:19.582000 - ECD2*49 (+0.07s)
2016-01-20 06:00:19.587000 - ECD1*59 (+0.01s)
2016-01-20 06:00:19.593000 - EME400/B6 (+0.01s)
2016-01-20 06:00:19.617000 - EFE100/1E (+0.05s)
2016-01-20 06:00:19.621000 - ESU511/1DX (+0.01s)
2016-01-20 06:00:19.626000 - ESU510/1DX (+0.01s)
2016-01-20 06:00:19.626000 - ESU510/1DX (+0.0s)
2016-01-20 06:00:19.657000 - EFD103/1E (+0.06s)
2016-01-20 06:00:19.657000 - EFD104/1E (+0.0s)
2016-01-20 06:00:19.676000 - ESU1.04*49 (+0.04s)
```

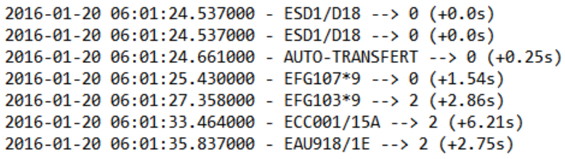
Figure 4.10: An extract of clustering results using the separation criterion based on average plus standard deviation of temporal differences of consecutive events

the timestamp $2016 - 01 - 2006 : 00 : 19.437$ of the reference device EMC001*9. In the window W after such timestamp an high number of events have been reported, and the overall average and standard deviation have been strongly influenced by them. It is interesting to take a look at the statistical measures $avg(E_c^i)$, $stdev(E_c^i)$, and their sum. The value of the standard deviation (2.08 seconds) is much higher than the average (0.37 seconds). This provides the robustness that the average based criterion lacks when the value of $avg(E_c^i)$ tends to 0, as we have seen in Figure 4.9.

Of course, using average plus standard deviation to separate the clusters does not exclude the possibility to use some multipliers k_1 and k_2 (with $k_1, k_2 \in \mathbb{Q}^+$) to change the weights of these two statistical measures. For example, we could compute the separation distance as $k_1 \cdot avg(E_c^i) + k_2 \cdot stdev(E_c^i)$. Unfortunately, there is no way to tell in advance which values of k_1 and k_2 we should choose, especially because they strongly depend on the interpretation that we give to the correlation between devices producing events in the same cluster. If we want to have just a few, big clusters with a separation distance that include in the same cluster events that are not regular, we should settle for high values (for example: from 2 to 5). Otherwise, if we believe that correlations between devices are very temporarily close to the preceding devices, we should use a low value of weights (for example: 1 or 1.5).

A consequence of using this clustering technique is that the correlation between two events e_1 and e_2, distant in time more than the value of $avg(E_c^i) + stdev(E_c^i)$, depends from the presence of other events happened between $e1$ and $e2$. If the same temporal distance between e_1 and e_2 was encountered in another E_c^i, the devices related to these two events would have been put into different clusters, because the temporal value that determines a division in cluster changes with respect to i. This occurs also for fixed distance clustering where the separation value doesn't depend on i, because long chains of events close in time to each other can create correlations between devices that otherwise wouldn't be recognized as correlated by the algorithm. The creation of such chains is actually not a problem in our context, because we are modelling the behaviour of anomalies and we expect that a fault could cause a sequence of other faults and irregularities.

Mean shift The Mean Shift algorithm is a mode-seeking algorithm for clustering. The reason of the name mode-seeking comes from the fact that these techniques search for the mathematical mode in a set of data. Mean Shift identifies all the modes (local maxima) in a surface derived from a kernel density estimate function [44]. The local maxima of the surface coincide with regions with a high density of data points. The number of local maxima, dynamically found by the algorithm, corresponds to the number of clusters. Each data point is part of the basin of attraction of one and only one maximum among all the maxima found by the algorithm. Points in the same basin of attraction are part of the same cluster [45]. In standard implementations of mean shift, like the one that we tested, points on the borderline of high density areas are not marked as noise. This can lead to confusing separations between events, like the one showed in Figure 4.11. In this image, the membership of an event to a cluster is denoted by an arrow followed by the number of the cluster (that starts from 0). As we can see, the algorithm separates the events of devices EFG107*9 and EFG103*9 that are 2.86 seconds one from the other, but it keeps in the same cluster ECC001/15A and EFG103*9 which are 6.21 seconds distant. In the interpretation of the correlations as a chain of temporal close events, EFG103*9 should be removed from cluster 2. This strange

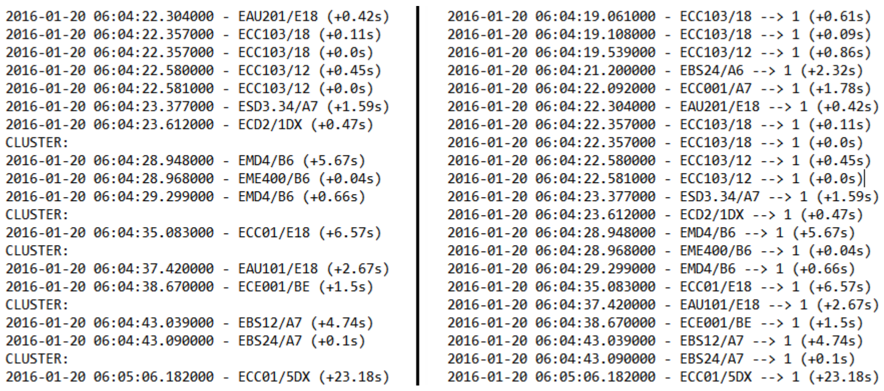


```
2016-01-20 06:01:24.537000 - ESD1/D18 --> 0 (+0.0s)
2016-01-20 06:01:24.537000 - ESD1/D18 --> 0 (+0.0s)
2016-01-20 06:01:24.661000 - AUTO-TRANSFERT --> 0 (+0.25s)
2016-01-20 06:01:25.430000 - EFG107*9 --> 0 (+1.54s)
2016-01-20 06:01:27.358000 - EFG103*9 --> 2 (+2.86s)
2016-01-20 06:01:33.464000 - ECC001/15A --> 2 (+6.21s)
2016-01-20 06:01:35.837000 - EAU918/1E --> 2 (+2.75s)
```

*Figure 4.11: A detail of the separation between clusters in an event sequence of device EMC001*9.*

behaviour is actually perfectly normal if we consider that the events shown in the figure are at the boundaries of clusters 0 and 2. A point in such position is not assigned to a cluster or another based on the distance to its immediate neighbours, but in which basin of attraction of the kernel density function estimate it lies into. For this reason, mean shift does not seem to be an effective algorithm to use in our setting. A direct comparison with the clustering based on average plus standard deviation shows that this last method provides a more meaningful separation between clusters. An example is shown in Figure 4.12

We can see how Mean Shift is not even able to separate the last event visualized in Figure 4.11, which is registered 23.18 seconds after



```
2016-01-20 06:04:22.304000 - EAU201/E18 (+0.42s)
2016-01-20 06:04:22.357000 - ECC103/18 (+0.11s)
2016-01-20 06:04:22.357000 - ECC103/18 (+0.0s)
2016-01-20 06:04:22.580000 - ECC103/12 (+0.45s)
2016-01-20 06:04:22.581000 - ECC103/12 (+0.0s)
2016-01-20 06:04:23.377000 - ESD3.34/A7 (+1.59s)
2016-01-20 06:04:23.612000 - ECD2/1DX (+0.47s)
CLUSTER:
2016-01-20 06:04:28.948000 - EMD4/B6 (+5.67s)
2016-01-20 06:04:28.968000 - EME400/B6 (+0.04s)
2016-01-20 06:04:29.299000 - EMD4/B6 (+0.66s)
CLUSTER:
2016-01-20 06:04:35.083000 - ECC01/E18 (+6.57s)
CLUSTER:
2016-01-20 06:04:37.420000 - EAU101/E18 (+2.67s)
2016-01-20 06:04:38.670000 - ECE001/BE (+1.5s)
CLUSTER:
2016-01-20 06:04:43.039000 - EBS12/A7 (+4.74s)
2016-01-20 06:04:43.090000 - EBS24/A7 (+0.1s)
CLUSTER:
2016-01-20 06:05:06.182000 - ECC01/5DX (+23.18s)
```

```
2016-01-20 06:04:19.061000 - ECC103/18 --> 1 (+0.61s)
2016-01-20 06:04:19.108000 - ECC103/18 --> 1 (+0.09s)
2016-01-20 06:04:19.539000 - ECC103/12 --> 1 (+0.86s)
2016-01-20 06:04:21.200000 - EBS24/A6 --> 1 (+2.32s)
2016-01-20 06:04:22.092000 - ECC001/A7 --> 1 (+1.78s)
2016-01-20 06:04:22.304000 - EAU201/E18 --> 1 (+0.42s)
2016-01-20 06:04:22.357000 - ECC103/18 --> 1 (+0.11s)
2016-01-20 06:04:22.357000 - ECC103/18 --> 1 (+0.0s)
2016-01-20 06:04:22.580000 - ECC103/12 --> 1 (+0.45s)
2016-01-20 06:04:22.581000 - ECC103/12 --> 1 (+0.0s)
2016-01-20 06:04:23.377000 - ESD3.34/A7 --> 1 (+1.59s)
2016-01-20 06:04:23.612000 - ECD2/1DX --> 1 (+0.47s)
2016-01-20 06:04:28.948000 - EMD4/B6 --> 1 (+5.67s)
2016-01-20 06:04:28.968000 - EME400/B6 --> 1 (+0.04s)
2016-01-20 06:04:29.299000 - EMD4/B6 --> 1 (+0.66s)
2016-01-20 06:04:35.083000 - ECC01/E18 --> 1 (+6.57s)
2016-01-20 06:04:37.420000 - EAU101/E18 --> 1 (+2.67s)
2016-01-20 06:04:38.670000 - ECE001/BE --> 1 (+1.5s)
2016-01-20 06:04:43.039000 - EBS12/A7 --> 1 (+4.74s)
2016-01-20 06:04:43.090000 - EBS24/A7 --> 1 (+0.1s)
2016-01-20 06:05:06.182000 - ECC01/5DX --> 1 (+23.18s)
```

Figure 4.12: On the left, a sequence of events separated in cluster with the criterion of average plus standard deviation. On the right, the same sequence separated with mean shift.

its predecessor in the log, an high value both per se and by considering the other temporal distances in the same in that sequence.

DBSCAN Another well-known clustering algorithm is DBSCAN: it falls under the density-based clustering algorithms and considers clusters as high density areas, separated by low density areas. The objective of this algorithm is similar to that of Mean Shift, although the implementation is very different. In order to find these areas, DBSCAN identifies the *core samples*, which are the data points that contain at least m other points in an ϵ radius. In our case, points are the events $e^{ik}_{d_j} \in E^i_c$ and the ϵ radius becomes a linear distance over the only dimension considered for this problem: the time. Points that are not core points but are contained in the radius of at least another core point are called *border points*. With respect to the Mean Shift algorithm, DBSCAN performs better in outlier detection, because the algorithm identifies anomalies by design. These points are called *noise points*, and are the points which are neither core nor border points [46]. Clusters are composed by core and border points which have the property of being density connected. The definition of density connectivity is

based on density reachability and direct density reachability. A pair of points p and q are density-connected if they are commonly density-reachable from another point o. A point p is density-reachable from a point q if there is a chain of points $p_1, \ldots, p_n$, with $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i. Finally, a point q is directly density-reachable from a point p if p is a core point and q is in p's ϵ-neighborhood, which is the set of all points within a radius of ϵ from the point p.
If we set $m = 1$, we obtain the same results we would get by using the fixed distance clustering with ϵ as the fixed temporal value to determine the clusters separation. This similarity suggests that DBSCAN maintains the same problem of that algorithm, which is the dependence from highly arbitrary parameters chosen by the user. Increasing m, the number of clusters decreases and the density of the clusters increases. In Figure 4.13 we show a result of DBSCAN application with m=2 and ϵ=12 seconds. The events denoted with a "-1" have been classified

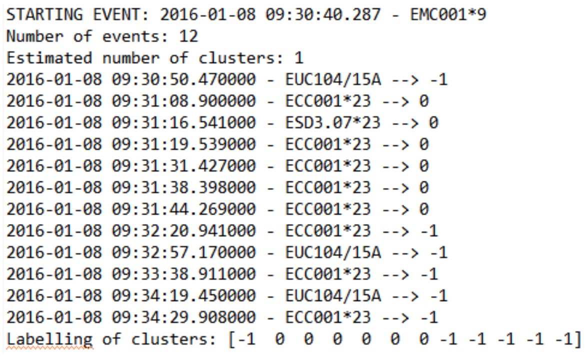

```
STARTING EVENT: 2016-01-08 09:30:40.287 - EMC001*9
Number of events: 12
Estimated number of clusters: 1
2016-01-08 09:30:50.470000 - EUC104/15A --> -1
2016-01-08 09:31:08.900000 - ECC001*23 --> 0
2016-01-08 09:31:16.541000 - ESD3.07*23 --> 0
2016-01-08 09:31:19.539000 - ECC001*23 --> 0
2016-01-08 09:31:31.427000 - ECC001*23 --> 0
2016-01-08 09:31:38.398000 - ECC001*23 --> 0
2016-01-08 09:31:44.269000 - ECC001*23 --> 0
2016-01-08 09:32:20.941000 - ECC001*23 --> -1
2016-01-08 09:32:57.170000 - EUC104/15A --> -1
2016-01-08 09:33:38.911000 - ECC001*23 --> -1
2016-01-08 09:34:19.450000 - EUC104/15A --> -1
2016-01-08 09:34:29.908000 - ECC001*23 --> -1
Labelling of clusters: [-1  0  0  0  0  0  0 -1 -1 -1 -1 -1]
```

*Figure 4.13: Results from the application of DBSCAN to device EMC001*9.*

as noise points and therefore we will consider them as itemsets of size equal to 1. In this case the other non-noise events are all density connected and they form a single cluster, defined by the group of events with the "0" label.
Even if DBSCAN behavies better than mean shift thanks to the noise handling, it is still too ideal in our context because it's a density based

clustering technique, which does not take into account the sequential distance of consecutive events. Moreover, as a consequence of the fact that the ϵ parameter is fixed, DBSCAN has problems in handling varying densities, a characteristic observable also in our data, for many E_c^{il}. Our algorithm presented in this section and based on average plus standard deviation is more suited to cope with this problem. Overall, the best algorithm found so far to perform this task is the one based on average plus standard deviation, which we will use to generate all the clustering-based networks of this document unless otherwise specified.

4.1.4 Adding new fields

As we have seen in our discussion on the duplicate problem, the information contained in fields like "tag" or "state" can be useful to eliminate the ambiguity of temporally-close events that refer to the same device. Another obvious advantage is that they provide specific information about the devices that we are correlating, and therefore they allow to specialize our analysis. For this reason, we found out that adding new fields is something that is useful to do *after* having already generated a Bayesian network with only simple devices as variables. This allows us at first to get an handle of the general correlation between two devices, so that later we can see if such correlation is present only when particular values of some fields appear, or if it is generally valid.
For example, we may find out that two devices d_1 and d_2 often report events in the same W intervals. Later, we can generate a network by attaching the "level of priority" to these devices. We then discover that every time d_1 reports an event of high priority, d_2 reports an event of medium priority; instead, when d_1 reports an event of any other priority level, d_2 does not generate any event.

Instead of building a set of candidate devices C_i, we now have to generate a set of pairs $C_i' \; \forall i \in T$, starting as usual from E_c^i. Each pair in C_i' is of the type $(d_j, f(e_{d_j}^{ik}))$, where i and k are generic indexes for timestamp and event, and f is a function in a subset of the *field functions*: $f \in F' \subset F$. This means that $f(e_{d_j}^{ik})$ represents an additional field of the log that is related to events of d_j. The variables in our network will now become pairs. An example of the training set built in this way is shown in the Equation (4.14), where we used the notation tag^i to denote a specific value assumed by the field "tag".

$$\mathbf{S} = \begin{array}{c} \\ i \\ i+1 \\ i+2 \end{array} \begin{array}{c} \begin{array}{cccc} (d_1, tag^9) & (d_2, tag^2) & (d_2, tag^3) & (d_3, tag^7) \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{4.14}$$

An immediate consequence is that now the same variable can appear in the Bayesian network more than once, but with different values of the field f associated. The main downside is that now networks will involve more variables. The more specialized the field we decide to add, the more variables are needed to show the same interactions between devices. In the event log illustrated in Section 3.2 there are four fields that could be used: "tag", "state", "description", and the discretized "level of priority". These four fields compose the set F'.

4.2 Model generation

4.2.1 Variables choice

Once we have generated the set C of candidate variables, we must use a criterion to extract the set V of variables that will actually be part of the Bayesian network. Many choices are possible, and they often depend on which aspect of the devices we want to focus on.

4.2.1.1 Manual selection and device exclusion

When we already know the name of some devices of which we want to find the correlations, we can manually fill the set V. If we chose the timestamps T by extracting them from multiple reference devices, we would have to select these reference devices as variables if we want to perform the analysis on their correlations. Whenever T is obtained from a single reference device, instead, we must remove the reference device from the set of candidate devices C, otherwise it would appear in every line of the training set and generate an obvious but biased correlation with every other device in the Bayesian network.

4.2.1.2 Occurrences and support

A simple criterion is to select devices that appear more frequently in E_c, that is, the devices with the highest number of events within W minutes of all the timestamps $i \in T$. The number of occurrences $occ(d)$ of each device d can be computed as:

$$occ(d) = \sum_i p_i(d) \tag{4.15}$$

where $d \in C$ and $p_i(d)$ is a function defined as:

$$p_i(d) = \begin{cases} 1 & d \in C_i \\ 0 & \text{otherwise} \end{cases} \tag{4.16}$$

After ranking the occurrences of the devices, we can pick the first m of them. This is called *occurrences* variable selection criterion. Alternatively, we can compute the support $sup(d)$ of each device $d_j \in C$ as:

$$sup(d) = \frac{\sum_i p_i(d)}{|T|} \tag{4.17}$$

and select the devices d_j that have a support $supp_{d_j}$ greater than a value s. With this last method, called *support* criterion, we can be sure to not exclude variables that most frequently appear after the timestamps $i \in T$.

4.2.1.3 Temporal criteria

In some instances we might be interested in devices that appear with a temporal regularity after - or before - the timestamps in T. To simplify, we will now assume we are building a network on what happens after the timestamps. For example, consider a device d_1 whose events are observed to be reported always 10 seconds after some specific timestamps of the set T. The timestamps after which the events of d_1 appear are part of the set P, that is such that $P \subset T$. Our suspicion is that device d_1 is bound to events in P in a temporal sense. To follow the direction of this consideration, we introduce two new variable selection criteria: *average* and *standard deviation*.

With the *average* criterion we select devices whose events, on average, appear closest to the timestamps i. In detail, we compute the average *avg* as:

$$avg(d) = \frac{\sum\limits_{e \in E_c, d = \text{device}(e)} \text{timestamp}(e) - i}{\sum\limits_{e \in E_c, d = \text{device}(e)} 1} \tag{4.18}$$

and then pick the m devices with lowest average, or devices with the average lower than a certain value s. This method, however, does not consider how spread the events are around the average. It's not a good choice if we are interested in selecting devices that when they appear in the log after a timestamp i they do it after a regular time interval, but it's the best choice if we want to find the most rapid devices to be influenced on average by events with timestamps in T. To keep the spread into account we can use the *standard deviation* criterion, which consists in computing the standard deviation *stdev* of each device:

$$stdev(d) = \sqrt{\frac{\sum\limits_{e \in E_c, d = \text{device}(e)} (\text{timestamp}(e) - avg(d))^2}{\sum\limits_{e \in E_c, d = \text{device}(e)} 1}} \tag{4.19}$$

and taking the m devices with lowest *stdev*, or the devices with *stdev* lower than a certain value s. With this criterion we can select variables that appear with regularity around the same instant of time, but without knowing if they will appear close or far from the timestamps in T. If we are interested in devices that regularly appear immediately after each $i \in T$, we can combine the last two approaches and take the devices d with lowest $avg(d) + stdev(d)$ (criterion of *average plus standard deviation*).

If temporal criteria based on average are chosen it is not wise to choose an high value for the analysis window W. If we are interested in finding devices that usually appear regularly after a short amount of time, increasing the value of W will cause a greater exposure to noise with a progressively lower probability of finding new correlations. The noise in this case are the occurrences of devices that usually appear regularly within W of the timestamps in T but that with the enlargement of the window could be found again past W, causing an unexpected increase in the average and standard deviation values. These new occurrences could lead to a non-negligible modification of the average and standard deviation values of the devices. However, the noise problem for this criteria becomes less important if an high amount of data is available and if strong temporal correlations are actually present.
A condition that must be applied to temporal criteria is that the selected devices need to have the number of occurrences above a minimum threshold. If this threshold is not sufficiently high, the criteria are biased towards choosing devices with low occurrences because it is easier to have low average and low standard deviation with fewer data. On the contrary, if the threshold is too high, the variables chosen will often be the same selected by the occurrence or support criterion, failing to find devices with the behaviour desired.

4.2.1.4 Confidence

We now introduce another criterion of variable selection that we call *confidence*. The idea is to select variables that, in the entire event log, appear more times in the analysis window of the timestamps $i \in T$ and less times in any other part of the log. This means that we are selecting variables that are strongly correlated with the timestamps. To find these variables we must compute the confidence $conf(d)$ for each device $d \in C$ as:

$$conf(d) = \frac{\sum\limits_{i \in T} p_i(d)}{\sum\limits_{e \in E, d = \text{device}(e)} 1} \tag{4.20}$$

where the function $p_i(d)$ has been defined in Equation (4.16). Note that since the numerator in Equation (4.20) is the number of times in which the device d has appeared at least once after every E_c^i and the denominator is the total number of events of device d in the entire log, the values of $conf(d)$ might never reach 1, even if the device d always appears near to the timestamps in T. Another possibility is to change the numerator to be the total number of times in which d was found within W from the timestamp. In this case, however, the confidence value can become higher than 1 because of the duplicate problem and can lead to give more weights to devices that have events often duplicated. Indeed, these considerations share many similarities with the two approaches to the duplicate problem presented in Section 4.1.3.1. For the rest of this document we will use the version of confidence in Equation (4.20). The devices chosen with the confidence criterion needs to have a value of the denominator of Equation (4.20) higher than a minimum threshold. The reason for this choice is similar to what we have said for the temporal criteria. In fact, it is easy to get high confidence results if the total occurrences of a device in the entire log are only a few, while it is harder to obtain an high confidence value if the denominator is an high number.

4.2.1.5 Variable selection with preprocessing variations

All the presented variable selection criteria can be used with the addition of new fields to variables as explained in Section 4.1.4. Instead of computing support, average, standard deviation, and confidence of single devices, we just have to compute these values for pairs of the type (device, additional field). If we selected the timestamps T using multiple reference devices, then we can not apply temporal criteria and

the confidence criterion since they are implicitly based on the presence of a single reference device.
In general, confidence and temporal criteria can benefit from this new variable representation. From a temporal point of view, coupling devices and additional fields can increase our chances of finding regular variables, i.e., variables that steadily appear in the same temporal position before or after the timestamps in T. It's instead harder to find devices that activates after regular temporal intervals for any value of their additional fields.

The variable selection criteria described in this section assume that clustering has not been applied. Both temporal and confidence criteria find devices with particular properties related to timestamps $i \in T$ and can still be applied to the itemsets generated by clustering methods. Clustering with the confidence criterion needs only a small modification to Equation (4.20), to avoid the use of $p_i(d)$ in the numerator, since this function is defined on the itemsets C_i that are not present when we perform clustering. This is only a formal rewriting, and the results obtained from the application of the Equation (4.22) and from the Equation (4.20) are the same. We define first the set dev_i as $d = \text{device}(e), e \in E_c^i$, in order to define the index $w_i(d)$ as:

$$w_i(d) = \begin{cases} 1 & d \in dev_i \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

The new confidence criterion formula for clustering is:

$$conf(d) = \frac{\sum_{i \in T} w_i(d)}{\sum_{e \in E, d = \text{device}(e)} 1} \tag{4.22}$$

So, if we choose to apply any clustering technique presented in Section 4.1.3.2 together with temporal or confidence criteria, the variables of the model V would be the same as those chosen without the application of clustering.

Instead, occurrence and support criterion with clustering techniques can be rewritten in two ways. The first way doesn't impact the variable selection process but only the training set S row values:

$$occ(d) = \sum_i w_i(d) \tag{4.23}$$

$$sup(d) = \frac{\sum_i w_i(d)}{|T|} \tag{4.24}$$

These are the new formulations for Equations 4.15 and 4.17, respectively, and they have to be used in the same way as described in Section 4.2.1.2. The other way to count occurrences and support exploiting clustering allows also to improve the variable selection mechanism and consists in computing $occ(d)$ and $sup(d)$ with respect to C_{il} instead of C_i. This alternative formula for $occ(d)$ is:

$$occ(d) = \sum_l p_l(d) \tag{4.25}$$

where $d \in C$ and $p_l(d)$ is a function defined as:

$$p_l(d) = \begin{cases} 1 & d \in C_{il} \\ 0 & \text{otherwise} \end{cases} \tag{4.26}$$

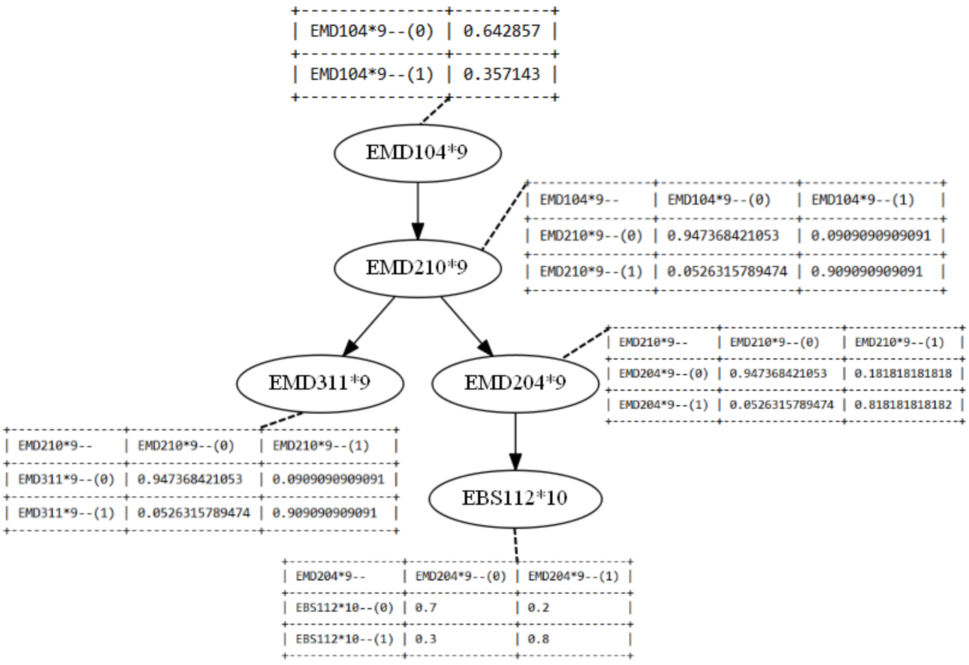
while $sup(d)$ is modified as:

$$sup(d) = \frac{\sum_l p_l(d)}{Q} \tag{4.27}$$

This second formulation is the one we have implemented in our work.

4.2.2 Network generation and interpretation

Once we have selected the variables V and we have built the training set S, we have everything is needed to generate structure and parameters of our network. In the networks that we will show in the rest of this document, we will always assume that we are using the support variable selection method, the training set generation described in Section 4.1.3, and that we are looking for correlations *after* the timestamps of a reference device, unless otherwise specified. Figure 4.14 shows a Bayesian network generated with the timestamps of the reference device EMC001*9, considering only reference events with priority L2, with duplicate elimination. This network graphically indicates that



```
+---------------+----------+
| EMD104*9--(0) | 0.642857 |
+---------------+----------+
| EMD104*9--(1) | 0.357143 |
+---------------+----------+
```

```
+---------------+-----------------+-----------------+
| EMD104*9--    | EMD104*9--(0)   | EMD104*9--(1)   |
+---------------+-----------------+-----------------+
| EMD210*9--(0) | 0.947368421053  | 0.0909090909091 |
+---------------+-----------------+-----------------+
| EMD210*9--(1) | 0.0526315789474 | 0.909090909091  |
+---------------+-----------------+-----------------+
```

```
+---------------+-----------------+-----------------+
| EMD210*9--    | EMD210*9--(0)   | EMD210*9--(1)   |
+---------------+-----------------+-----------------+
| EMD204*9--(0) | 0.947368421053  | 0.181818181818  |
+---------------+-----------------+-----------------+
| EMD204*9--(1) | 0.0526315789474 | 0.818181818182  |
+---------------+-----------------+-----------------+
```

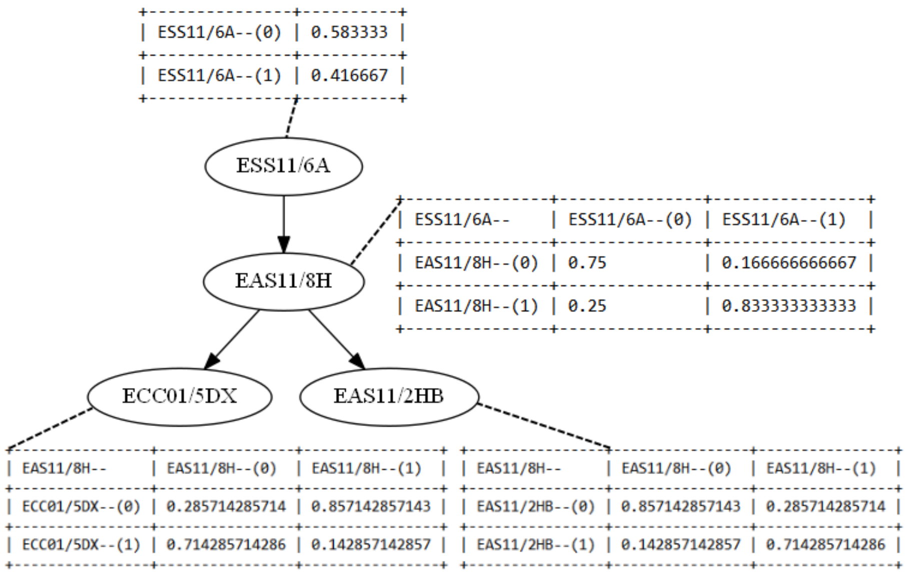
```
+---------------+-----------------+-----------------+
| EMD210*9--    | EMD210*9--(0)   | EMD210*9--(1)   |
+---------------+-----------------+-----------------+
| EMD311*9--(0) | 0.947368421053  | 0.0909090909091 |
+---------------+-----------------+-----------------+
| EMD311*9--(1) | 0.0526315789474 | 0.909090909091  |
+---------------+-----------------+-----------------+
```

```
+----------------+---------------+---------------+
| EMD204*9--     | EMD204*9--(0) | EMD204*9--(1) |
+----------------+---------------+---------------+
| EBS112*10--(0) | 0.7           | 0.2           |
+----------------+---------------+---------------+
| EBS112*10--(1) | 0.3           | 0.8           |
+----------------+---------------+---------------+
```

*Figure 4.14: The Bayesian network based on the reference device EMC001*9 with priority L2 and no duplicates.*

there are four correlations between the devices. To understand the strength of such correlations, we must look at the CPDs of the nodes (pointed with a dotted line). For example, from the CPDs we see that the probability of EMD311*9 being 1 given that its parent EMD210*9 is 1 is around 91%. We also notice that the CPD of EMD210*9 is

equal to the one of EMD311*9, which suggests us that these two devices (and EMD104*9) probably appear always together. The CPD of node EMD104*9, since it has no parents, shows the probability of seeing or not seeing an event of that device in an itemset, i.e., after the timestamps of the reference device, inside the analysis window W. Of course, CPD tables offer useful information, but they do not tell us everything immediately. For example, we do not know which is the probability of seeing device EMD104*9 when we have seen EMD311*9. By performing probabilistic inference on the network we obtain the answer of this question: 83.1%.
Now consider the network in Figure 4.15. Here we notice that the re-



```
+---------------+----------+
| ESS11/6A--(0) | 0.583333 |
+---------------+----------+
| ESS11/6A--(1) | 0.416667 |
+---------------+----------+
```

```
+---------------+---------------+----------------+
| ESS11/6A--    | ESS11/6A--(0) | ESS11/6A--(1)  |
+---------------+---------------+----------------+
| EAS11/8H--(0) | 0.75          | 0.166666666667 |
+---------------+---------------+----------------+
| EAS11/8H--(1) | 0.25          | 0.833333333333 |
+---------------+---------------+----------------+
```

```
+----------------+----------------+----------------+
| EAS11/8H--     | EAS11/8H--(0)  | EAS11/8H--(1)  |
+----------------+----------------+----------------+
| ECC01/5DX--(0) | 0.285714285714 | 0.857142857143 |
+----------------+----------------+----------------+
| ECC01/5DX--(1) | 0.714285714286 | 0.142857142857 |
+----------------+----------------+----------------+
```

```
+----------------+----------------+----------------+
| EAS11/8H--     | EAS11/8H--(0)  | EAS11/8H--(1)  |
+----------------+----------------+----------------+
| EAS11/2HB--(0) | 0.857142857143 | 0.285714285714 |
+----------------+----------------+----------------+
| EAS11/2HB--(1) | 0.142857142857 | 0.714285714286 |
+----------------+----------------+----------------+
```

Figure 4.15: The Bayesian network based on the reference device ESS11/5H with priority L1 and no duplicates.

lation between devices ECC01/5DX and EAS11/8H is not one of the correlations we are looking for. In fact, by looking at the CPD of the node ECC01/5DX we see that the probability of it being 1 while its parent is 1 is quite low, around 14.3%. So we deduce that the edge between these two nodes does not represent any correlation. Indeed, the two devices almost seem to be in a mutually exclusive relationship. Even if it might sound counterintuitive at first, it is clear that an edge in the Bayesian network does not imply correlation (in the sense defined by our problem) because the Bayesian model can not grasp the meaning of the difference between the values "1" and "0", and just looks for patterns between variables. Despite this consideration, the network is far from useless. Inference can still be used to find the most likely answer of the appearance (1) or not appearance (0) of the devices in the window W. Graphically, however, the network structure becomes much less reliable. We will propose a solution to this problem in Section 4.3.1.

Suppose now that we have a network like the one in Figure 4.14, with edges that actually represent strong correlations between nodes. We now ask ourselves which is the the meaning of these edges. Does an arrow from device EMD104*9 to EMD210*9 imply that the first is the *cause* of the other? Or does it mean that the first usually appear after the other? The answer to both of these question is no. Because of the way in which the training set is built, that is, without considering the temporal order of appearance of the devices, the resulting network provide no meaning in a causal sense. What we can say when looking at a strong correlation is that, for example, EMD210*9 appears with high probability when also EMD104*9 has appeared. Or, more in detail, an event of EMD210*9 appears with probability 91% within 5 minutes of an event of EMC001*9 of priority L2 when also an event of EMD104*9 has appeared. Moreover, if we do not perform inference on specific pairs of variables, we can not say anything about variables that are not directly connected with an edge. This problem will also be addressed in the next section.

4.3 Post-processing

4.3.1 Inference labels

As we explained in the previous section, an edge between two devices does not necessarily imply correlation, at least in the context of our problem. However, by looking at the CPDs and performing inference we can deduce which variables are correlated. Since our models must be interpreted directly by an user, to make the Bayesian networks more meaningful we present a graphical variation based on inference. In Figure 4.16 we show a Bayesian network with the introduction of labels on the edges. We have also marked strong correlations in red. We call these labels *first inference labels*, because they show the result of particular inference queries over the nodes. In detail, given two devices

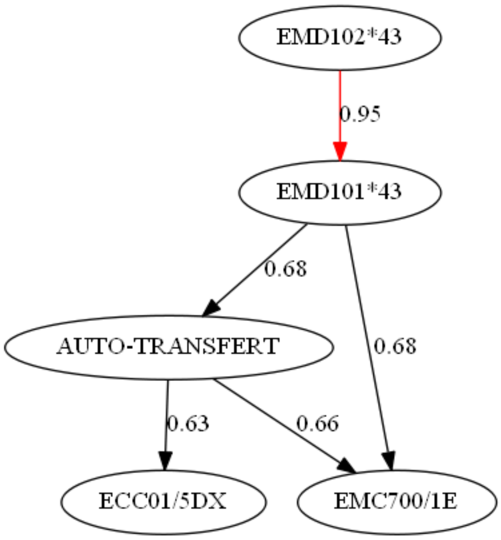


*Figure 4.16: The Bayesian network of device EMC001*9, with priority L1, no duplicates and the addition an inference label on the edges.*

d_1 and d_2 and an edge e that connects them, with d_2 being the parent of d_1, the first inference label shows the result of the probabilistic query:

$$p(d_1 = 1 \mid d_2 = 1) \tag{4.28}$$

In our context, these labels tell us which is the probability of seeing an event of the device d_1 when we have already seen an event of d_2. They are equivalent to the CPD probabilities of having the node value equal to 1 when the parent is 1, and for this reason we could also call them *CPD labels*. In fact, since the most important information given by the CPDs in our problem are the values in the cells whose variables in rows and columns take only the "1" values, it makes sense to graphically represent only that probability. The networks represented in this way provide the information in which we are interested in a much more immediate way. The CPDs are still used to perform inference and conditional probabilities which involve some "0" as values of variables can still be discovered with it, if necessary.
These CPD labels, however, shows us the probability of an event only in one direction. To check which is the probability of the parent when an event of the son has been observed, we must compute the following query:

$$p(d_2 = 1 \mid d_1 = 1) \tag{4.29}$$

The result of these new queries can be attached to the edges of the network, separated by the symbol "|" from the first inference labels. We call these new labels *second inference labels* and we say that networks like the one in Figure 4.17 has the graphical addition of *double inference labels*. The double inference labels allow to understand if a correlation is more relevant in one direction or the other. For example, it might happen that when device d_2 appears, d_1 rarely reports events; instead if d_1 appears then d_2 always appears. In this case we could be tempted to say that most likely d_1 is the cause of d_2. Indeed, in some instances we might be correct, assuming that the event of a device does not have multiple causes. However, it is easy to see that this is not true in general. Imagine that we know for sure that there is a device d_3 that causes the independent appearance of d_1 and d_2, and another device d_4 that causes only the activation of d_2. Suppose that

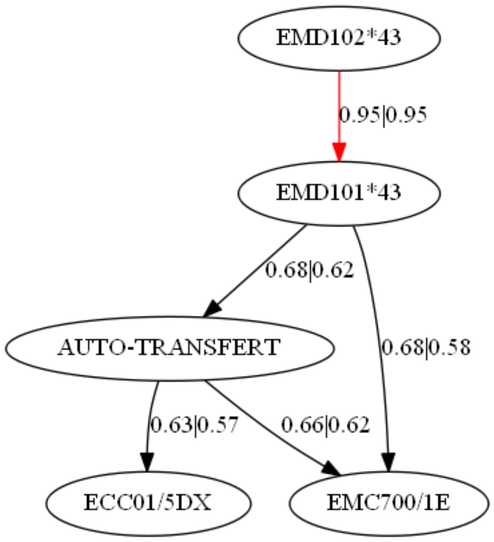


*Figure 4.17: The Bayesian network of device EMC001*9, with priority L1, no duplicates and the addition double inference labels on the edges.*

d_3 and d_4 were not included in the network because of the variable selection criteria we choose. Now, we finally understand why d_1 and d_2 are always seen together, and we also understand why the appearance of d_2 (when caused by d_4) does not imply that we will see d_1. This problem has been briefly discussed in Section 2.2.1, where these hidden variables have been called *latent variables*. Moreover, because of the non-temporal nature of our training set, we do not know if a device will appear before or after another correlated device. In the previous example, there are no constraints that impose the appearance of d_1 before d_2 or vice versa, and our reasoning with respect to d_3 and d_4 remains valid either way.
At the light of these considerations we will generally consider our Bayesian networks as non causal, and therefore we will not perform intervention queries, which we have discussed in Section 2.2.2.

4.3.2 Reference device

When we use a reference device to extract the timestamps, our network assumes a different meaning. We are no longer finding device correlations before or after some key time instants, but after or before a specific device has generated an event, which is correlated to all the devices in the network, even if such device does not appear in the network. In this case it could be interesting to visually show such reference device, in order to immediately identify its relationship with the rest of the network. Figure 4.18 shows the same network in Figure 4.14 with the addition of a node with blue edges that represents the reference device and its connections with the other nodes. We call

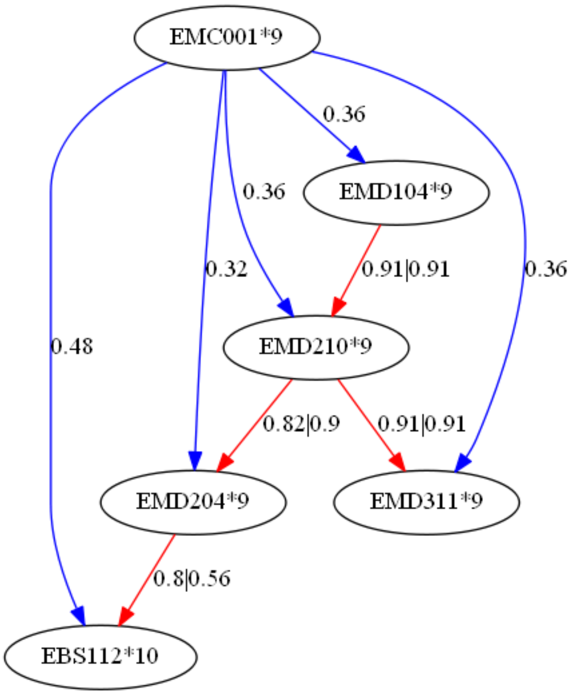


*Figure 4.18: The Bayesian network of device EMC001*9, with priority L2, no duplicates, the double inference labels on the edges and the reference device.*

the node of device EMC001*9 *reference node*. Obviously it is not part of the Bayesian model and it provides, with labels on the blue edges, the probability of seeing all the other devices in the network after its events. For example, the probability of seeing an event of EBS112*10 after an event of EMC001*9 is 48%.

4.3.3 Locations

There are many ways in which the Bayesian model can be graphically altered in order to integrate additional information found in the event log. We tried this approach with the fields “H0” and “H1”: we showed “H0” as a rectangular box enclosing a subgraph and labeled with the locations of the devices, and “H1” by coloring the nodes belonging to that location and showing the correspondence in a legend. The intention behind this graphical addition is to allow the user to quickly identify interesting correlations, like relationships between far away devices in different areas that should not interfere with each other, or to simply localize the position of correlated devices.

4.3.4 Devices information

An interesting information that is related to the validity of the found correlations is the number of times in which each device is present in the training set. In Figure 4.20, we added this information in a box around each node. As we can see, this network has a fairly low number of occurrences. The inference labels, however, tell us that between the devices there are good correlations that are quite unlikely to be all due to noise. Anyway, by looking at the occurrences an user could decide how much does she want to trust the network. Beside occurrences, another useful addition to our networks is the average and standard deviation of the difference between the timestamps at which the events of the devices appear with respect to the timestamp of the initial event of the reference device. These values are the same as those illustrated in Section 4.2.1.3. They can be used to graphically evaluate the temporal placement of the devices, for example after choosing a

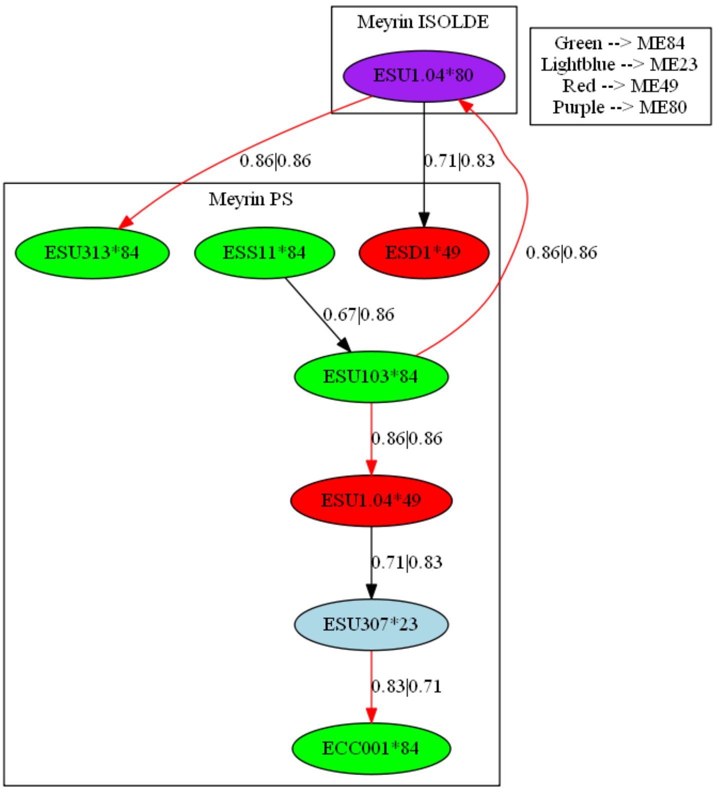


*Figure 4.19: The Bayesian network of device ESS1*84, with priority L1, no duplicates, the double inference labels and the device locations.*

temporal variable selection criteria, or they can prompt the generation of a new network based on these criteria. In Figure 4.21 an example of these additional *device information* is shown.

4.3.5 Inference network

Since the correct interpretation of our Bayesian networks is strongly based on the result of inference queries, we could argue that all that really matters is the inference. Following this consideration we built

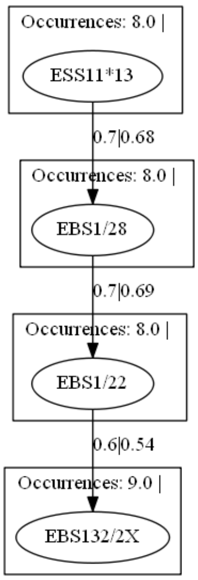


Figure 4.20: The Bayesian network of device EXS106/2X, with priority L1, no duplicates, the double inference labels and the occurrences.

a type of network, that we call *inference network*, in which every connection between every pair of variable is possible, provided that at least one of the two inference labels computed between those pairs is greater than a given parameter. Changing such parameter allows us to generate inference networks with an high or low threshold for the relevance of the correlations we want to visualize. Figure 4.22 shows the inference network of the Bayesian model in Figure 4.14. In this case, we used a relatively high threshold in order to avoid showing too many edges. One of the advantages of this representation is that the user can immediately identify which appearances of devices can lead to other highly likely appearances, without having to manually perform the inference queries. In other situations, instead, we might have generated Bayesian networks with an high number of variables, which normally turn out to be hard to evaluate graphically. In this cases we

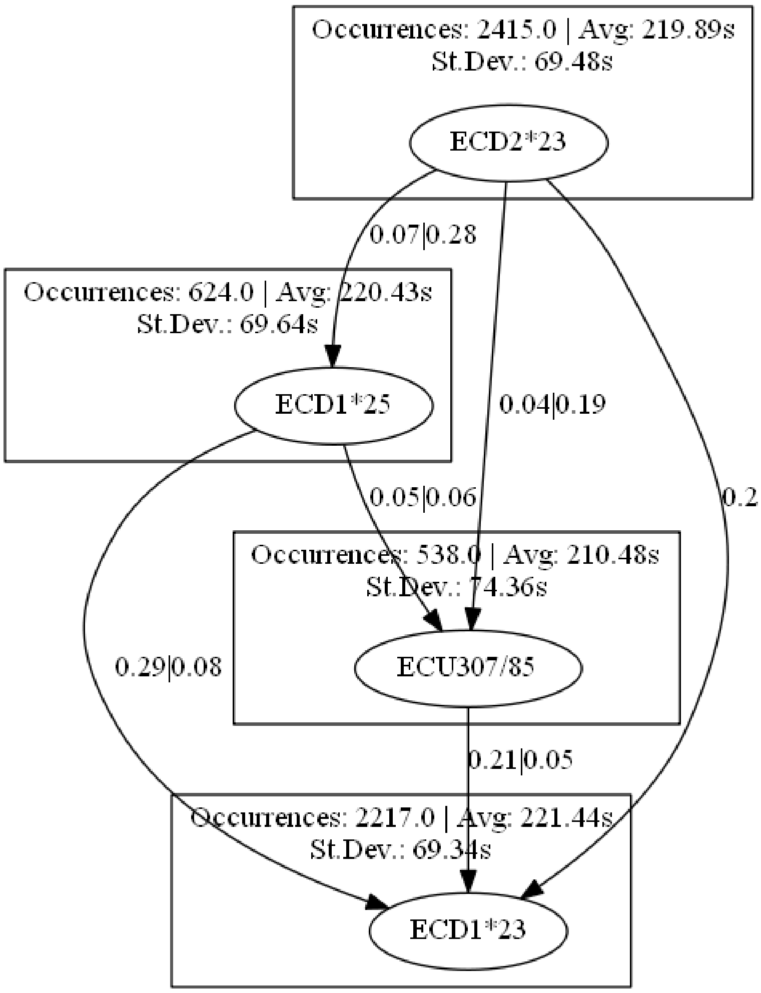


*Figure 4.21: The Bayesian network of device ECD1*62, with priority L1, no duplicates, the double inference labels and the devices information.*

can reduce the number of edges shown by using an inference network that filters out all the edges with low values of inference labels.

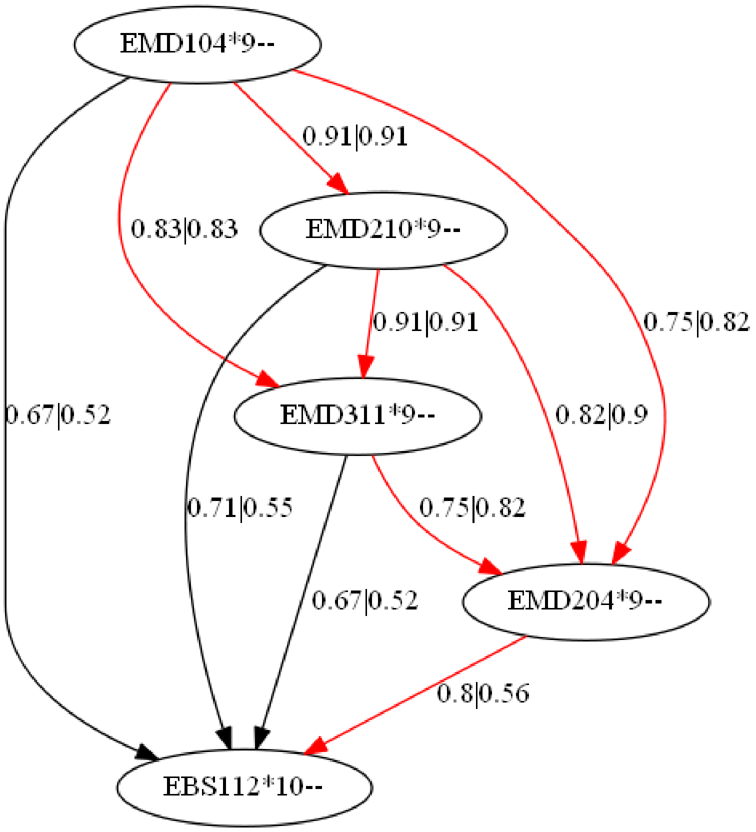


*Figure 4.22: The inference network of device EMC001*9, with priority L2 (without duplicates).*

Chapter 5

The Markov chain model

This chapter illustrates the work done on the Markov chain model with a structure that mirrors that of Chapter 4. As we will see, many techniques used in preprocessing and postprocessing of Bayesian networks are applicable also in Markov chains. In Section 5.1 we deal with the preprocessing phase, in Section 5.2 we explain the interpretation of this model, in Section 5.3 the possible graphical additions. A summary of the methods presented in this chapter will be reported in the Table 6.1 in Section 6.1.

5.1 Preprocessing of data

5.1.1 Timestamps and analysis window

The selection of the set of timestamps T for Markov chains can be performed by taking into account the same choices that we discussed in Section 4.1. We can extract T starting from a reference device, a set of multiple reference devices, or we can directly obtain it from a human expert. The same considerations also hold for the analysis window, which we still consider set to $W = 5$ minutes.

5.1.2 Training sequences generation

In order to generate the Markov chain model, we must first generate the sequences of states that we will use in the training phase to learn the transition probabilities. Assuming that we already selected, among all the candidate devices C, the set of states $V \subset C$ that we want to represent in the model (such choice is addressed in Section 5.2.1), we now have to define the *training sequences*. Each training sequence is a vector $\vec{P_i}$ of variable length associated to a specific timestamp $i \in T$, where each element of $\vec{P_i}$ is a device $d_j \in V$.

To build a sequence $\vec{P_i}$ in the case in which we want to model what happens *after* the timestamps, we must first consider the event set E_c^i, scan all the events $e \in E_c^i$ in chronological order from i to $i + W$ and save in a vector $\vec{B_i}$ the devices d_j such that $d_j = \mathit{device}(e)$. Devices d_j are saved into $\vec{B_i}$ keeping the chronological order of the events to which they refer to. By repeating this procedure for each $i \in T$ we obtain the multiset $B = \{\vec{B_i} | i \in T\}$, which admits possible identical sequences. The criteria to select the states to save in V are applied to this multiset.

The major differences between a set C_i, generated in the preprocessing phase for the learning of Bayesian networks, and a vector $\vec{B_i}$, are that the latter admits the presence of duplicate devices and takes into account the ordering of its elements.

Once we have at disposal the set V, the training multiset $P = \{\vec{P_i} | i \in T\}$ is generated by filtering the sequences $\vec{B_i} \in B \ \forall i$, eliminating all the occurrences of the devices $d_j \notin V$. The filtering procedure does not modify the relative ordering of the devices which form $\vec{P_i}$.

Equation (5.1) shows an example of training sequences of the multiset P where $V = \{d_1, d_2, d_3, d_4, d_5\}$ and $T = \{1, 2, 3\}$. We notice that sequences have a variable length that simply depends from how many events of the devices in V were present after the timestamps inside the analysis window W.

$$\begin{aligned} P_1 &= \begin{bmatrix} d_2 & d_4 & d_2 & d_4 \end{bmatrix} \\ P_2 &= \begin{bmatrix} d_1 & d_3 & d_5 \end{bmatrix} \\ P_2 &= \begin{bmatrix} d_1 & d_3 & d_4 & d_5 & d_4 & d_5 \end{bmatrix} \end{aligned} \tag{5.1}$$

If we include in V all the devices of our dataset there would be no need to apply variable filtering: the multisets B and P would be equal and the generated model will represent direct transitions probabilities. In this case, sequences of adjacent devices in vectors $\vec{B_i}$ would correspond to the devices with temporally consecutive log entries in the database. Unfortunately it is impossible to visualize hundreds or thousands of variables and their transition probabilities in a graph, but if there is no need to visualize the entire state diagram, it could be feasible to learn a model with a huge number of variables. As we have seen in Section 2.3, the training of a Markov chain is not computationally expensive. As reported in Pomegranate's documentation [41], there are no latent factors to train and so no expectation maximization or iterative algorithms are needed to train anything. For each sequence, maximum likelihood estimates are used to update the parameters of the distributions. The calculation of the probability of a sequence of variables is fast too, because it consists in the computation of simple products. This would be a way to get information from the model when we cannot visualize it.

However, what commonly happens it is that the analysis performed on a graphically displayable set of variables V (i.e., a low number of variables) generally shows *relative* transitions probabilities. The sequence of variables inside $\vec{P_i}$, in fact, represents the relative ordering of the variables in V inside the corresponding vector $\vec{B_i}$, while the vector $\vec{B_i}$ contains the true, unfiltered ordering. This discrepancy can lead to misinterpretations of the transition probabilities. We will call this problem *the filtering problem.*

These problems, however, do not decrease the usefulness of the Markov chain model. If we generate a Bayesian network and a Markov chain based on the same subset of variables V, we can discover information about the temporal relative order of activation of two variables d_1 and d_2 that are strongly correlated in the Bayesian network. If by looking at the Markov chain we see that the probability of the transition from the state corresponding to d_1 to the state corresponding to d_2 is much higher than the probability of the inverted transition from d_2 to d_1, we gain a much more reliable information about the causality of the relationship than what we could discover from the same situation with a Bayesian network. We will show in Section 6.6 an example of a Bayesian network and a Markov chain applied on the same data and subset of variables. Moreover, suppose that there exists a frequent activation of events of two devices d_1 and d_2, where d_2 registers its events usually 20 seconds after d_1. It is possible that in many cases this correlation is not recognized by the Markov chain which considers all the variables in the log, because other devices could have registered events that have little or no relation to the frequent process regarding d_1 and d_2. An analysis based on the relative temporal ordering of devices could instead highlight better the presence of this frequent pattern.

5.1.2.1 Duplicates

The duplicate problems that we have seen in the Bayesian model can be present also in the training sequences of Markov chains, but lead to different consequences. We consider the timeline we used to illustrate the duplicate problem in Section 4.1.3.1, which we report again in Figure 5.1 for better clarity. If we do not modify our algorithm to handle the duplicates, we obtain the sequences in Equation (5.2):

$$\begin{aligned} \vec{P_i} &= \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix} \\ \vec{P_{i+1}} &= \begin{bmatrix} d_2 & d_3 & d_4 \end{bmatrix} \end{aligned} \tag{5.2}$$

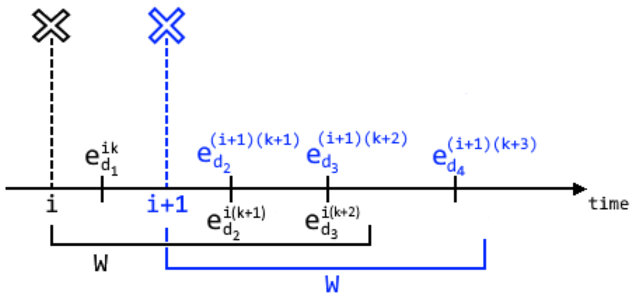


Figure 5.1: An example of timeline with some events related to two timestamps i (in black) and $i+1$ (in blue).

Since, as explained in Section 2.3, Markov chains do have the Markov property, what determines the transition probability to a device (i.e., a state) is just the device the immediately before it in the sequence. In Equation (5.2) the pair d_2, d_3 is present twice, instead of only one time as the timeline shows. This means that the transition probability from d_2 to d_3 is higher than it should be.
Let's now consider the same sequences generated by removing the duplicates (that is, by considering each event only once):

$$\begin{aligned} \vec{P_i} &= \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix} \\ \vec{P_{i+1}} &= \begin{bmatrix} d_4 \end{bmatrix} \end{aligned} \tag{5.3}$$

Now the problem is that we lose the transition between d_3 and d_4. We can lose at most one transition for each pair of timestamps i and $i+1$, that is, we may lose in total a maximum of $|T|-1$ transitions. In Markov models with a total number of transitions considerably greater than $|T|-1$, the impact of such loss is minimized. In comparison, in the Bayesian networks the approach based on duplicate elimination can make us lose up to $|V|-1$ correlations (where each correlation is a pair of variables both set to 1 in the same training instance) for each pair of consecutive timestamps, for a total (in the worst case) of $(|V|-1)\cdot(|T|-1)$.

Moreover, there is a way to completely eliminate all the disadvantages when we remove duplicates in Markov chains. If we detect that we are going to lose a transition because of the overlaps between two timestamps, we can add such transition as a single sequence in the training sequences. In the example, we just have to add $[d_3\ d_4]$. Equivalently, we could also merge the two sequences and obtain a single sequence $[d_1\ d_2\ d_3\ d_4]$. This new approach can be defined as "perfect solution" and every Markov chain shown in the following has been generated in this way. The latter "perfect" implementation based on the sequence merging is preferable to the former, because we have to consider that the vectors $\vec{B_i}$ to which we are referring to in these examples will be filtered to create the training sequences $\vec{P_i}$. If only one device between d_3 and d_4 is not selected as a variable of the model V, the addition of the sequence $[d_3\ d_4]$ will not change any probability of the model to be learned. If instead, for example, d_1, $d_4 \in V$ and d_2, $d_3 \notin V$, the sequence merging approach provides to the training procedure a new example of transition from d_1 to d_4.

5.1.2.2 Clustering

As we explained in the previous section, the relevant information in a Markov chain is given by consecutive devices. By applying clustering, we divide the sequences in multiple, shorter sequences. This can be done with the same metodologies explained in Section 4.1.3.2. Besides the set C_{il} that defines the candidate variables of each cluster $l \in \{1, q_i\}$, we define the vector $\vec{B_{il}}$ and the corresponding filtered vector $\vec{P_{il}}$.

Breaking a sequence related to a timestamp $i \in T$ in q_i parts means that we are losing $q_i - 1$ transitions during the learning process. The best case scenario would be the situation in which we manage to break the sequences in the correct position that separates two groups of events that have nothing to do with each other.

Clustering also helps with the filtering problem, since it naturally separates long chains and can avoid the situation in which two devices

that are temporally far away are consecutive after the filtering of the vector $\vec{B}_i$.
Conservative applications of clustering which tend to create few sequences containing many devices could be a good option for the Markov chain setting. For example, a choice could be the criterion based on average and standard deviation.

5.1.2.3 Adding new fields

The addition of new fields in the log can be done in a way very similar to the one described in Section 4.1.4. Following the same approach, we build a new set C'_i that replaces C_i and contains pairs of the type $(d_j, f(e^{ik}_{d_j}))$. To build the set $\vec{B}_i$, as done in Section 5.1.2, we have to scan all the events $e \in E^i_c$ in chronological order from i to $i+W$, and save the pairs $(d_j, f(e))$ such that $d_j = \mathit{device}(e)$. Of course, the sequences obtained at the end of the training sequences generation will be composed by pairs too. This addition allows to specialize the Markov chain in finding transitions between precise states of the devices. As for Bayesian networks, the four fields “tag”, “state”, “description”, and “level of priority” can all be used freely, with the same considerations.

5.2 Model generation

5.2.1 The choice of states

The selection of states for the set V follows the same principles and methods in Section 4.2.1. First of all, the occurrences of the devices $d \in C$ can be computed as $occ(d)$:

$$occ(d) = \sum_{i \in T} \sum_{k=0}^{|\vec{B}_i|-1} b(\vec{B}_i[k], d) \tag{5.4}$$

where $b(d_j, d)$ is a Boolean function that compares two devices:

$$b(d_j, d) = \begin{cases} 1 & d_j = d \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

The definition in Equation 5.4 allow us to use the *occurrences* method in this model. The support criterion, instead, is not defined for Markov chains. Instead, we introduce the *frequency* criterion, which computes $\mathit{freq}(d)$:

$$\mathit{freq}(d) = \frac{occ(d)}{|T|} \tag{5.6}$$

$\mathit{freq}(d)$ can assume values greater than 1 because the numerator in Equation (5.6), differently from the support criterion in the Bayesian model, does count multiple appearances of the same device for each timestamp $i \in T$. The result of the frequency method, however, is conceptually similar to the support: it allows to select devices that exceed a relative frequency threshold.
The temporal criteria can be applied exactly as explained in Section 4.2.1 because they do not operate on sequences, but directly on the set E_c. The same holds for the confidence criterion.

We now introduce a new criterion called *close pairs* that allows to select devices that frequently appear one after the other. First of all we define a function $pairs(d_a, d_b)$ that returns the number of times in which two devices d_a and d_b were found one next to the other in the sequences:

$$pairs(d_a, d_b) = \sum_{i \in T} \sum_{k=0}^{|\vec{B}_i|-2} z(\vec{B}_i[k], \vec{B}_i[k+1], d_a, d_b) \tag{5.7}$$

where the function $z(d_j, d_h, d_a, d_b)$ is defined as:

$$z(d_j, d_h, d_a, d_b) = \begin{cases} 1 & (d_j = d_a \ AND \ d_h = d_b) \ OR \\ & (d_j = d_b \ AND \ d_h = d_a) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

Now we compute *pairs*(d_a, d_b) for all the devices $(d_a, d_b) \in D \times D$ with $d_a \neq d_b$, and we rank the results. The criterion consists in taking all the devices that appear in the m pairs with most occurrences. This method inherently looks for devices that have a strong temporal order of activation and generates transitions that represents more truthfully what happens in terms of event reporting in the log. As explained in Section 5.1.2, the filtering problem of the vectors $\vec{B_i}$ that produces $\vec{P_i}$ causes sequences to lose most of their elements and alters the overall structure of the chain by putting as consecutive two devices that might have been separate by other events. The main way to avoid this is to select an high number of states; this method instead tries to bypass the problem by selecting devices that are immediately adjacent in the log and that will result in *real* direct transition among them.
However, it is not guaranteed that all the chosen devices will appear in adjacent positions in the log. For example, we may select a pair of devices d_1, d_2 that often appear at the beginning of the analysis window W, and a pair d_3, d_4 that usually appear at the end of W. If these are the only variables we select, our sequences will inevitably contain transitions between elements of the first pair to elements of the second pair, even if they are temporally afar. This is just another consequence of the filtering problem and we could also try to resolve it performing clustering. Anyway, the transitions between devices of different frequent pairs should have low probabilities because the number of transitions between elements of the same pair have high occurrences by design. In the best case in which all the frequent pairs come from a chain of activations that involves different devices, we would exploit at best the advantages of the close pairs criterion.

As we did for Bayesian networks, these state selection criteria can be applied with the addition of new fields and clustering, with trivial modifications of the equations that mirror the ones in Section 4.1.4.

5.2.2 State diagram generation and interpretation

After the selection of the variables V and the application of the filtering process, we are ready to train the Markov chain. The multiset of sequences P is the only input needed to learn a Markov chain. In the networks that we will show in the rest of the document, we will always assume that we are looking for correlations *after* the timestamps of a reference device and that when we apply clustering we use the method based on average plus standard deviation, unless otherwise specified.

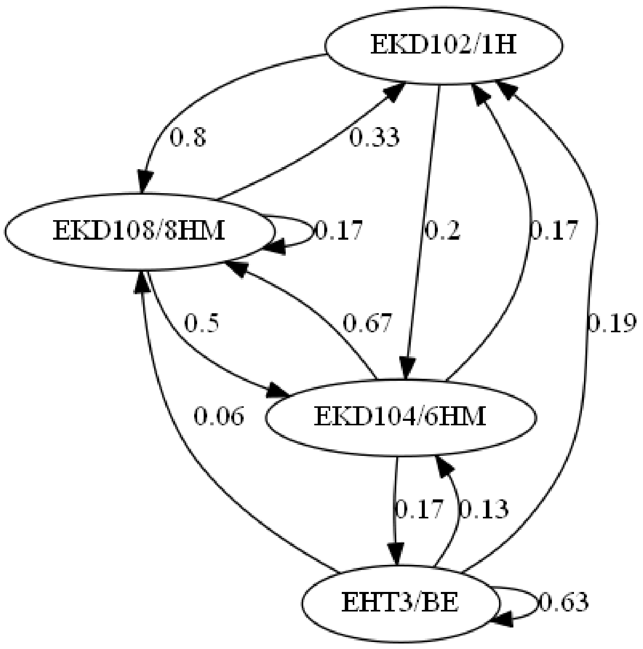


Figure 5.2: Markov chain based on the reference device EHS60/BE with priority L1 using confidence criterion.

The network in Figure 5.2 shows four devices selected with the confidence criterion. We can verify that the learning algorithm of the Pomegranate library creates consistent Markov chains because the sum of all the outgoing transition probabilities is 1 for every state. In some cases the sum is almost 1 only because of the approximation to the

second digit used for a clearer visualization. The possible presence of sequences of the same devices lead to the creation of self-loops. For example, EHT3/BE has an high probability on its self-loop.
When no self-loops are present we can deduce that there is not even an example in the training multiset P which presents consecutive appearances of the same devices. It would be interesting to perform an analysis considering states with additional fields, to discover the order of activation of tags or descriptions related to events of EHT3/BE.
Some conclusions that we can draw from the state diagram in Figure 5.2 are that after seeing an event of device EKD102/1H is very likely, within the window W, to see an event of device EKD108/8HM, after which we have some probability to see EKD104/6HM and then again EKD108/8HM, in a loop with probabilities equal to 50% and 67%. On the contrary, if we see EHT3/BE we most likely will keep seeing events of the same device.

5.3 Postprocessing

As we did for the Bayesian model in Section 4.3, we can extend the graphical representation of Markov chains to include additional information that can help with our analysis.

5.3.1 Arc coloring and edge filtering

With more than 4 variables, too many edges are drawn and the visualization like the one in Figure 5.2 is no more clear. In Figure 5.3 we represent the same network of Figure 5.2 without drawing the edges with a probability lower than 0.15 and coloring in red the edges with a probability greater than 0.5. With this choice, the network loses the property of Equation (2.10) in Section 2.3, but we can find more easily frequent activations of devices, like the arc from EKD102/1H to EKD108/8HM which has an high probability (0.8).

If we decide to visualize more variables, the minimum threshold for drawing an edge could be increased. We have tried to increase it to

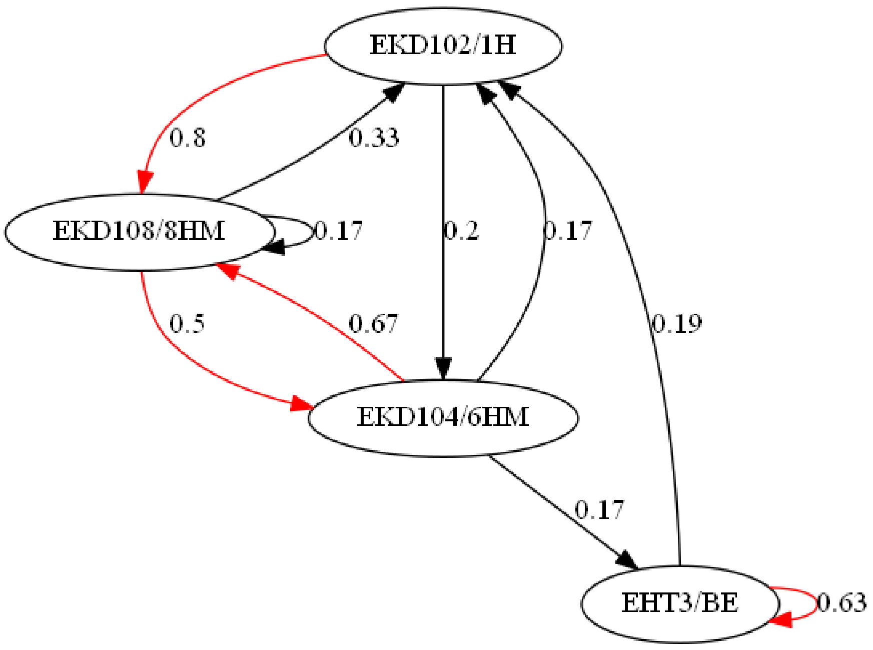


Figure 5.3: Markov chain based on the reference device EHS60/BE with priority L1 using confidence criterion, with red arcs and 0.15 as minimum threshold for edge visualization.

reach the value 0.33, which we suggest as a maximum value for the threshold. We suppose that if a device d_1 is followed by another device d_2 more than a third of the times, this shouldn't be considered as pure noise but it could be a significative correlation.

5.3.2 Temporal labels

Since Markov chains provide a way to observe the sequentiality between events reported by some devices, we realized that it could be useful to visualize the temporal distance between every pair of device in the network. The state diagram in Figure 5.4 shows a Markov chain where, after the transition probability on each edge, are reported between parenthesis two values in seconds. Consider an edge that goes from a

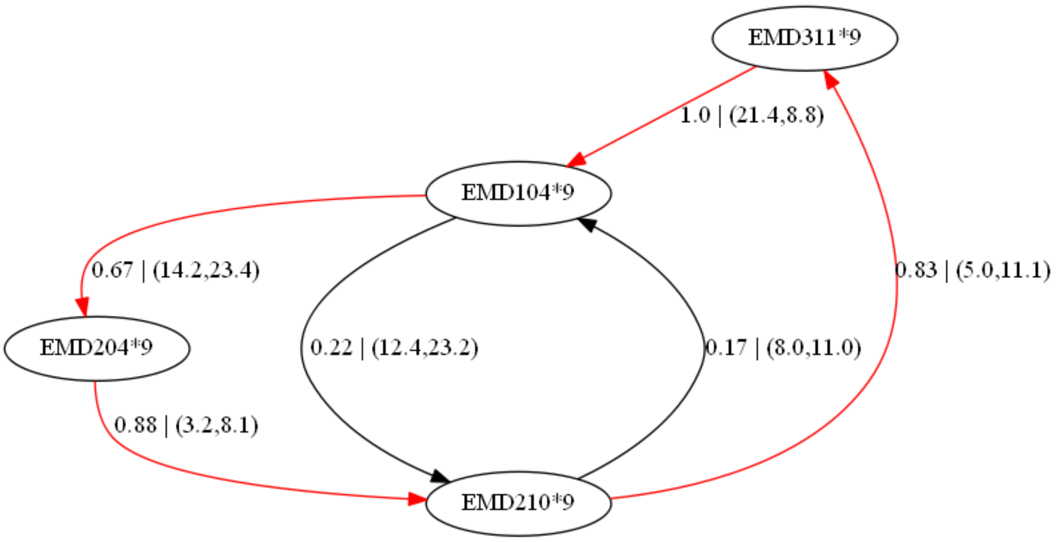


*Figure 5.4: Markov chain with temporal labels based on the reference device EMC001*9 with priority L2, using temporal criterion based only on minimum standard deviation*

device d_1 to another device d_2. The first value in the parenthesis is the average time of the appearance of the events of d_2 after the events of d_1, while the second value is the standard deviation of the same time of appearances.

States in Figure 5.4 have been selected with the temporal criterion based on minimum standard deviation with respect to the timestamps of events of the reference device EMC001*9, with priority L2. We notice how the values of average and standard deviation of the time elapsed between two events of adjacent devices in a sequence are low too: the average ranges from a minimum of 3.2 seconds to a maximum of 21.4 seconds and the standard deviation from 8.1 to 23.4 seconds. The transition probability values are generally high and we can spot a cycle in the diagram which involves only red arcs with just one of the four arcs having a probability lower than 0.8. However from this representation we cannot know which is the first and last device of such frequent cycle. There are many reasons to believe that we could have found a significant correlation but we have to know on how many occurrences this network is based on to be confirmed.

5.3.3 Locations and device information

The addition of locations and information about the occurrences, average, and standard deviation to the state diagram is straightforward and follows the same considerations done in Section 4.3.3 and Section 4.3.4. In Figure 5.5, we generate the same network in Figure 5.4 with additional statistics for every state. We can now verify if the correlations found in Figure 5.4 are based on a sufficient number of data. In each box that surrounds a device d we can also see how many seconds, after the events of the reference device, the events of device d appear in average ("Avg"), and the standard deviation ("St.Dev.") of the time interval of the same appearances. We expect the latter to be relatively low since the represented devices has been chosen with the criterion of minimum standard deviation. In this Markov chain, the minimum occurrence threshold to take into consideration the selection of a device as a state was set to 5 occurrences. We find out that the strong correlations we found are actually based on few data, but the similar values of occurrences, average, and standard deviation suggest the possible presence of dependencies among these devices.
Figure 5.6 shows the value of the fields "H0" and "H1" for each state, with the same graphical conventions seen for Bayesian networks, applied to the same network of Figure 5.4. From this image we discover that all the four devices share the same area "H0" ("Meyrin Jura"), and sub-area "H1" ("ME9").

5.3.4 Reference device

Another graphical addition is the reference device, as we did already in Bayesian networks. However, here the edges that connect the reference device node to the other states show labels that do not represent the

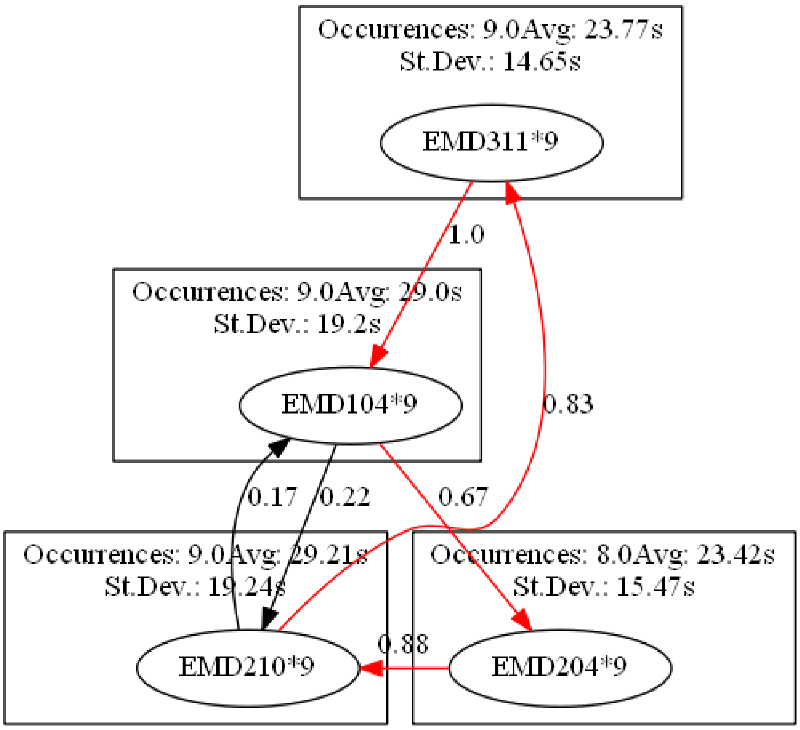


Figure 5.5: Markov chain in Figure 5.4 with additional information.

support of such states, but the probability of seeing a state as the *first* after the event of the reference device, or as *last* before the event of the reference device. In Figure 5.7 we show the Markov chain based on the same network in Figure 5.4 with the reference device. As we can see, that EMD104*9 is the first device of the training sequences $\vec{P_i} \in P$ for 64 % of the cases. When it is not the first device of a sequence, it always appears after EMD311*9. We underline that a device that is the first appeared in the sequence $\vec{P_i}$ may not necessarily be the first device in the correspondent unfiltered sequence $\vec{B_i}$.

Figure 5.8 shows a Markov chain that represents what happens before the timestamps of the reference device EHS60/BE with priority

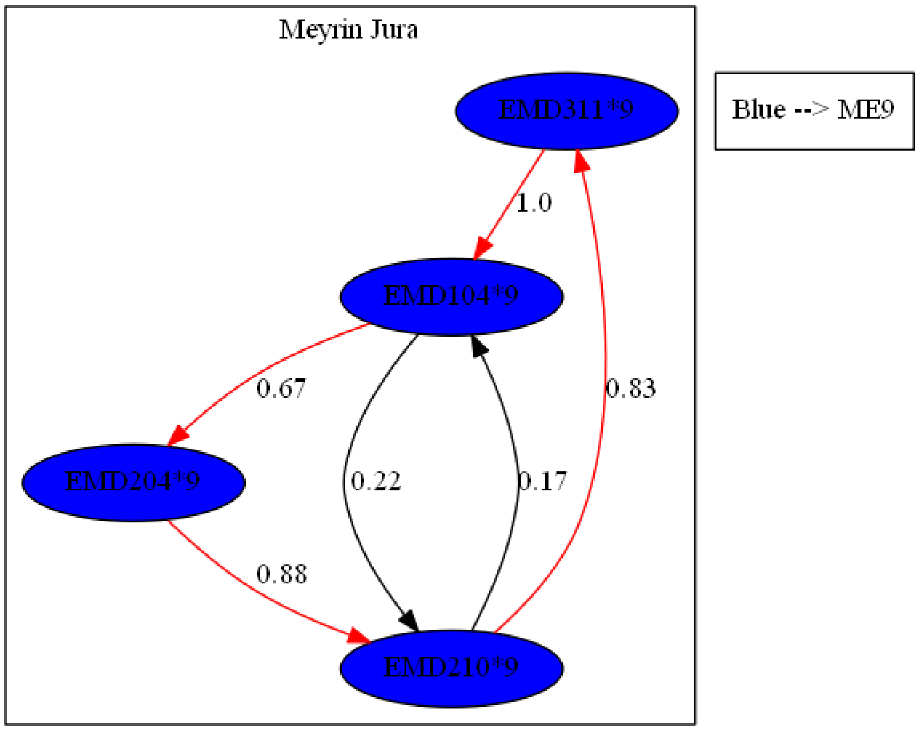


Figure 5.6: The Markov chain in Figure 5.4 with the addition of the locations of devices.

L3. In this case, we see that EHD33/BE is the only device to report an event before EHS60/BE. This means that EHD33/BE is also a sink for the network: every sequence $\vec{P}_i$ terminates with this device.

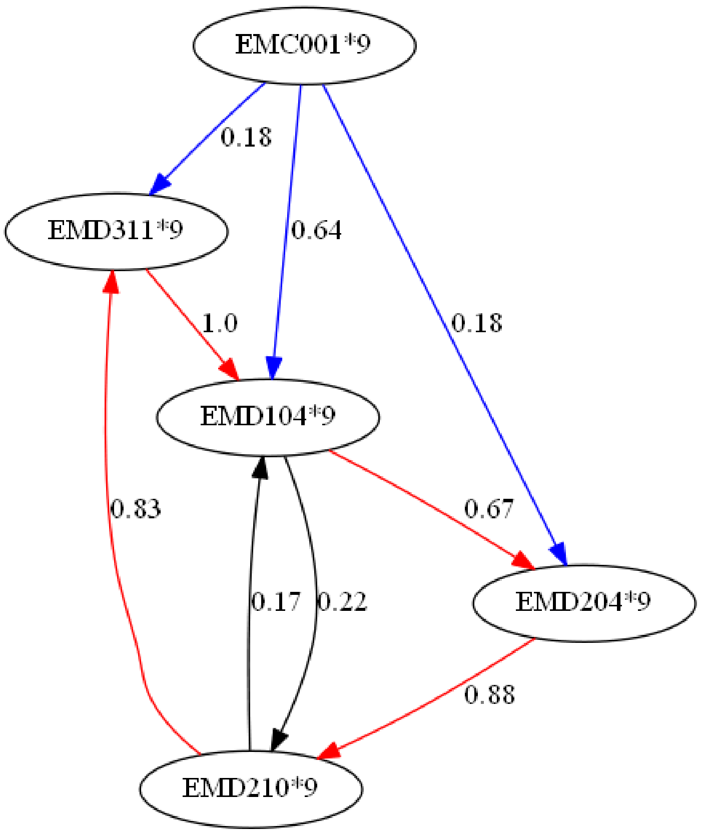


Figure 5.7: The Markov chain in Figure 5.4 with the addition of the reference device.

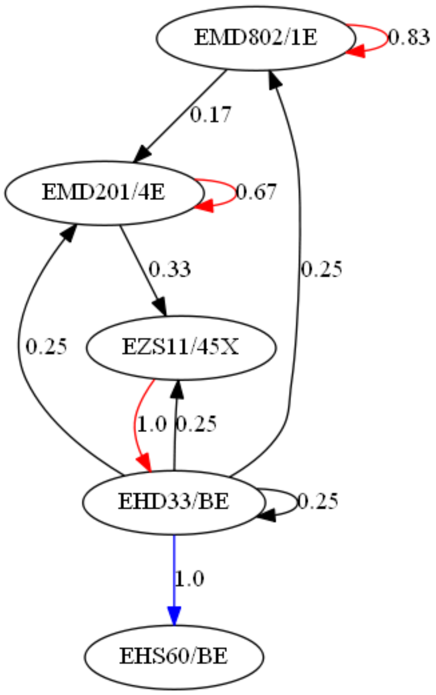


Figure 5.8: The Markov chain related to what happens before the events with priority L3 of the reference device EHS60/BE, with the graphical addition of the reference device.

Chapter 6

Examples of use

In this chapter we show some Bayesian networks and Markov chains from which we can draw interesting conclusions. First, in Section 6.1 we summarize the main methods presented in Chapter 4 and Chapter 5. In Section 6.2 we show how much, in practice, the presence or absence of duplicates can influence our Bayesian networks. In Section 6.3 we provide a comparison of models built with different variable selection criteria, while in Section 6.4 we show and comment some models with additional fields attached. In Section 6.5 we discuss about the value of the analysis window W. In Section 6.6 we compare the results of the Bayesian model and Markov chains with a practical example. In Section 6.7 we show an example of a network based on multiple reference devices and discuss its utility.

6.1 Summary of methods

Before showing some examples of use of our models, we summarize the main methods presented in Chapter 4 and Chapter 5. In Table 6.1 we show such techniques, split up in the three parts: preprocessing, model generation, and postprocessing.

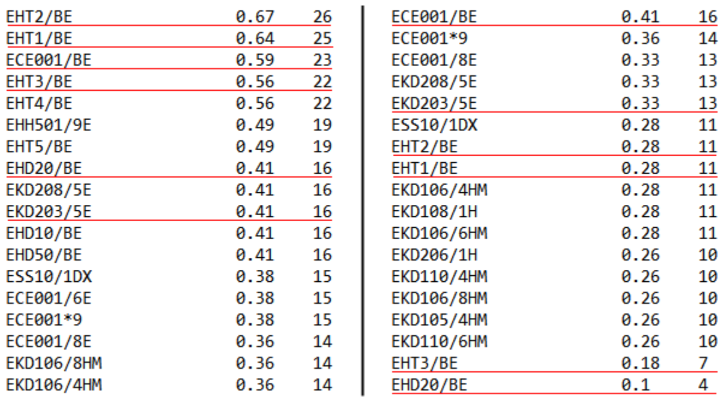
Phase	**Method**	**Section (BN)**	**Section (MC)**
Preprocessing			
Timestamp selection	Manual	4.1.1	5.1.1
	Reference device	4.1.1	5.1.1
	Multiple reference devices	4.1.1.1	5.1.1
Training set or training sequences generation	Standard	4.1.3	5.1.2
	Duplicate handling	4.1.3.1	5.1.2.1
	Clustering	4.1.3.2	5.1.2.2
	Additional fields	4.1.4	5.1.2.3
Model generation			
Selection of states and variables	Occurrences	4.2.1.2	5.2.1
	Support	4.2.1.2	×
	Frequency	×	5.2.1
	Temporal	4.2.1.3	5.2.1
	Confidence	4.2.1.4	5.2.1
	Close pairs	×	5.2.1
Postprocessing			
Graphical additions	Inference labels	4.3.1	×
	Reference device	4.3.2	5.3.4
	Temporal labels	×	5.3.2
	Locations	4.3.3	5.3.3
	Device information	4.3.4	5.3.3
Alternative representation	Inference network	4.3.5	×

Table 6.1: Summary of the main methods for preprocessing, model generation and post processing.

Next to each method we indicate the section in which it was explained for the Bayesian networks (BN) and Markov chains (MC). Any cell containing a cross in the “Section” columns indicates that the corresponding method cannot be used with that model (BN or MC).

6.2 The influence of duplicates

To understand the impact of the duplicates on Bayesian networks we show some examples of how the number of occurrences of candidate variables changes with respect to the removal of duplicates. In Figure 6.1 we see on the left a list of candidates variables found with the duplicates, and on the right the list without duplicates. Both lists are related to the timestamps extracted from the reference device EHS60/BE, with priority L1. Besides each device, the first number indicates the support with respect to the reference device; the second number indicates the number of occurrences in the training set. The two lists are partial rankings and they show only the top positions. As we can see, even



EHT2/BE	0.67	26	ECE001/BE	0.41	16
EHT1/BE	0.64	25	ECE001*9	0.36	14
ECE001/BE	0.59	23	ECE001/8E	0.33	13
EHT3/BE	0.56	22	EKD208/5E	0.33	13
EHT4/BE	0.56	22	EKD203/5E	0.33	13
EHH501/9E	0.49	19	ESS10/1DX	0.28	11
EHT5/BE	0.49	19	EHT2/BE	0.28	11
EHD20/BE	0.41	16	EHT1/BE	0.28	11
EKD208/5E	0.41	16	EKD106/4HM	0.28	11
EKD203/5E	0.41	16	EKD108/1H	0.28	11
EHD10/BE	0.41	16	EKD106/6HM	0.28	11
EHD50/BE	0.41	16	EKD206/1H	0.26	10
ESS10/1DX	0.38	15	EKD110/4HM	0.26	10
ECE001/6E	0.38	15	EKD106/8HM	0.26	10
ECE001*9	0.38	15	EKD105/4HM	0.26	10
ECE001/8E	0.36	14	EKD110/6HM	0.26	10
EKD106/8HM	0.36	14	EHT3/BE	0.18	7
EKD106/4HM	0.36	14	EHD20/BE	0.1	4

Figure 6.1: Comparison of candidate variables with duplicates (on the left) and without duplicates (on the right) for the reference device EHS60/BE with priority L1.

in a network with a relatively low number of variables, there might be a considerable number of duplicates. Occurrences of devices like EHT1/BE and EHT2/BE are more than doubled, while most of the other device occurrences are increased. Moreover, some devices in one of the two rankings are not present in the other, which shows how the duplicates can alter the variables we pick with this criterion.

Now we generate two Bayesian networks on the 6 devices underlined in red and compare results in the case with duplicates and without duplicates. The two networks are shown in Figure 6.2. Consider the

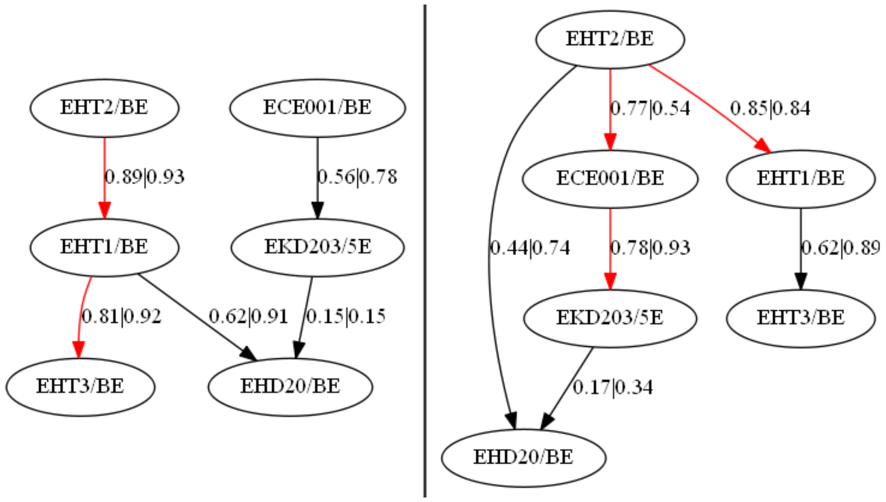


Figure 6.2: Two Bayesian networks related to the device EHS60/BE: one (on the left) with duplicates and the other (on the right) without duplicates.

relationship between devices EHT1/BE, EHT2/BE and EHT3/BE. In both networks, these devices are connected by the same edges, but the probabilities on the inference labels are higher with the duplicates. As we discussed in Section 4.1.3.1, making some correlations stronger is one of the consequences of keeping the duplicates. However, there is also the possibility that the network without duplicates is under-representing such correlations. Either way, the strong correlations between these devices are evident in both cases.
Now focus on the devices ECE001/BE, EKD203/5E and EHD20/BE. Again, they show the same relationship among them in both networks. The difference is that now the inference values without duplicates are

always higher than the ones with duplicates. This means that most likely the network with duplicates is weakening these correlations by duplicating some rows and making some of these devices appear in training rows alone, without correlations. Indeed, this very situation happens between EHD20/BE and EKD203/5E and was shown in Figure 4.4 of Section 4.1.3.1.

6.3 The use of different variable selection criteria

Not all the criteria explained in Section 4.2.1 and Section 5.2.1 lead to good models for all the timestamps sets T. For example, consider the Bayesian network in Figure 6.3, built according to the reference device ERD15*45 with priority L2. This network was generated by choosing the devices with the highest number of occurrences. As the figure shows, no relevant correlations were found, therefore the dependencies between the nodes are not based on a high probability of the simultaneous presence of the devices in the itemsets, but on the other CPD values. For example, the inspection of the CPD shows us that when ETZ755*43 is 0, ECC001*3 is almost always 0. All the other correlations in the network are of the type 0 - 0 as well, which is the pattern the Bayesian model is trying to reproduce with connections between the nodes in figure. We may suppose that there are no regular correlations at all with respect to this reference device. However, if we change variable selection method we may find something interesting. In Figure 6.4 we found a single correlation between devices EBD1.01*50 and EBD1.09*50, using the confidence variable selection method. By selecting the devices that report more events within the analysis window of ERD15*45 than in the rest of the log, we accidentally found two devices with a low standard deviation and very similar average. The number of occurrences is not very high, but the temporal regularity of these events could be considered enough to suppose that this correlation is valid.

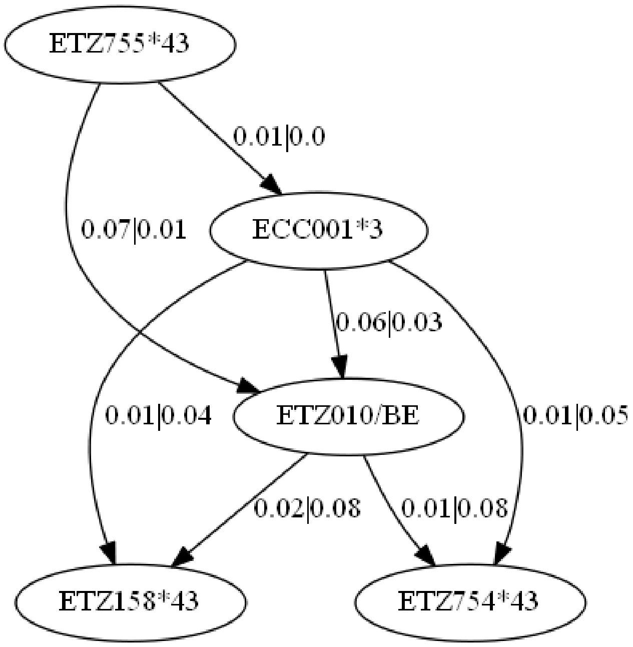


*Figure 6.3: Bayesian network of reference device ERD15*45, with priority L2 and variable selection method based on the occurrences.*

The next example shows the application of different variable selection criteria for Markov chains. In Section 5.2.1 we have defined the close pairs criterion, which can be utilized exclusively for the Markov chain model. We now show the Markov chain based on the reference device EHS60/BE with priority L1 obtained with the close pairs criterion (Figure 6.5), and we compare it to the one obtained with the occurrences criterion (Figure 6.6). Both figures consider events happened not more than 5 minutes after the corresponding timestamp of the reference device, and arcs with probability lower than 0.25 have not been represented. These Markov chains, as well as all the others in the rest of this section, have been generated using clustering for the sequences generation. The clustering method chosen is based as usual on average plus standard deviation and the states of the network present

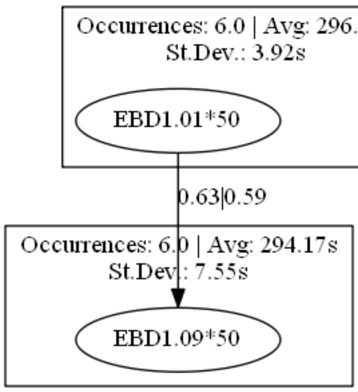

Occurrences: 6.0 | Avg: 296.73s
St.Dev.: 3.92s
EBD1.01*50
0.63|0.59
Occurrences: 6.0 | Avg: 294.17s
St.Dev.: 7.55s
EBD1.09*50

*Figure 6.4: Bayesian network of reference device ERD15*45, with priority L2 and variable selection method based on the confidence.*

the additional field “tag”. For every state, the name of the device and the value of the additional field are separated by “--”.
We see that the close pairs criterion manages to find interesting transitions probabilities that links the states EBS1/65 -- A15, EBS1/56 -- A30 and EBS1/56 -- A15. Those three devices are not selected by the occurrences criterion because they all have only 11 occurrences. The cycle related to the three states links similar tags and device names. Both the tag values start with "A" followed by a number and the names of the devices (EBS1/65 and EBS1/56) differs only in their numerical part. The more variables we choose to represent, the more these two criteria give us similar networks, because the occurrences criterion increasingly takes into account less frequent devices which could have interesting patterns.

We have also said in Section 5.2.1 that the close pairs criterion generally creates networks where there are groups of devices highly related among them, with little relation among states of the different groups. This is a consequence of the design and implementation of the criterion. The effect is not particularly visible in these two networks because many states with variable names that starts with “EHH” are selected

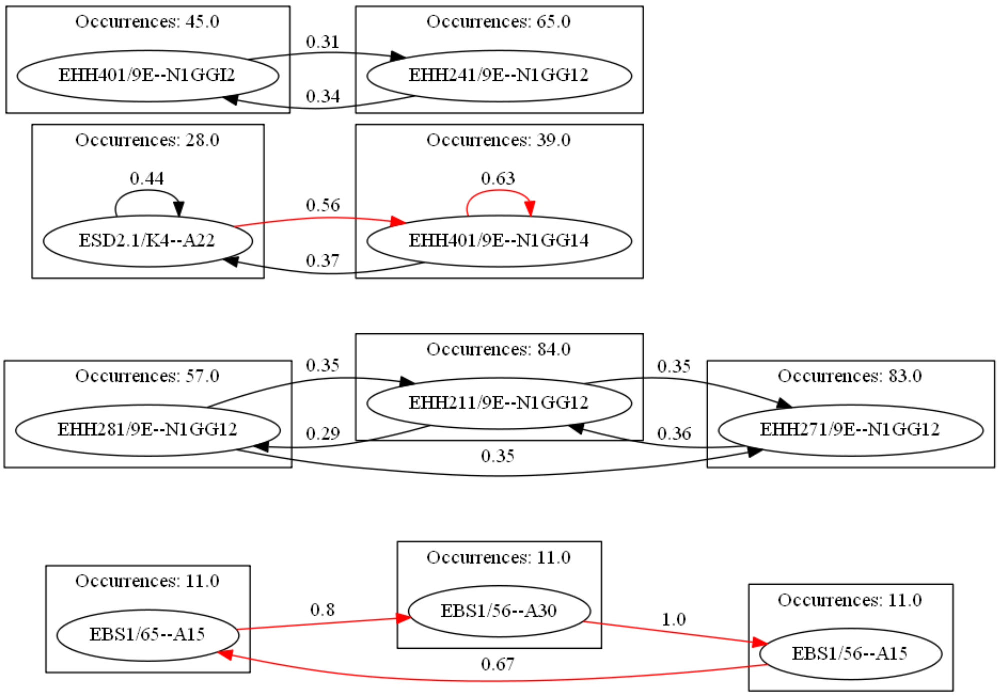


Figure 6.5: Markov chain related to the device EHS60/BE with priority L1, with states selected with the close pairs criterion and "tag" as additional field.

by both criteria. The group of "EHH" devices with their tags have many more occurrences compared to the other devices not selected by the occurrences criterion. Even if there is not a strong pattern in their consecutive appearances they manage to be at the top of the ranking of pairs for the close pairs criterion. A possible solution to this problem is discussed in Section 7.2, proposing a new way to rank pairs based on confidence.

We can spot in Figure 6.5 two distinct groups: the one with the states EBS1/65 -- A15, EBS1/56 -- A30 and EBS1/56 -- A15, and the other group composed of all the other devices. The latter group is not completely connected because we have set an high minimum threshold for edge visualization (0.25).

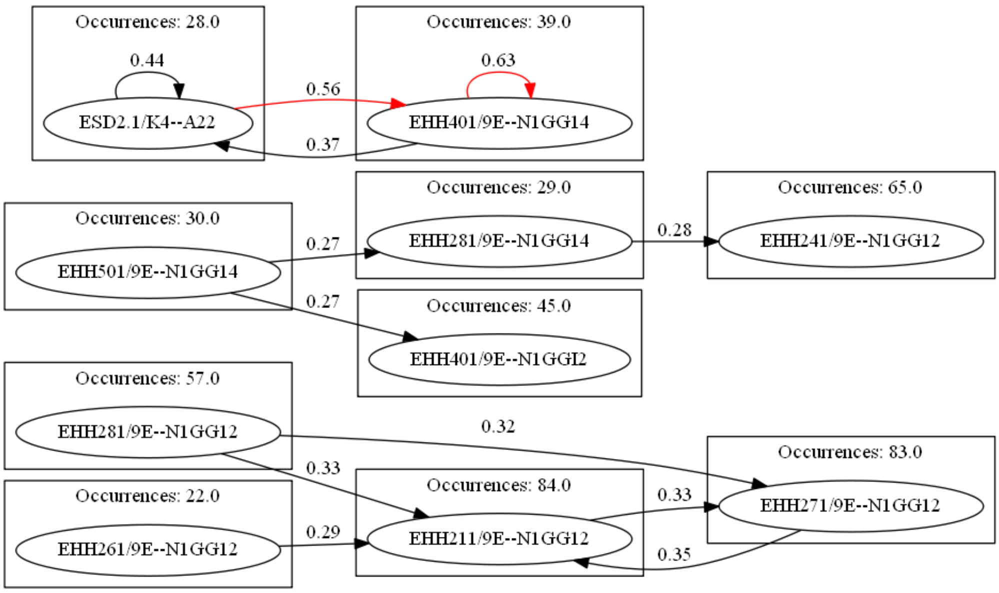


Figure 6.6: Markov chain related to the device EHS60/BE with priority L1, with states selected with the occurrences criterion and "tag" as additional field.

Most of the networks shown up to this point describe what happens *after* the events of a reference device. It can also be useful to analyze events happened right *before* a set of timestamps, to discover which devices may have been responsible for an abnormal behaviour, and Figure 6.7 is an example of such networks. It is based on a threshold value equal to 0.33 and shows how Markov chains becomes more and more disconnected as the threshold value increases; the result resembles a collection of rules. The network has been generated with the occurrences criterion by selecting a total of 150 states, with the addition of the "tag" as additional field. The probabilities shown in this Markov chain are close to describe the real adjacency relations between consecutive rows in the log (as we anticipated in Section 5.1.2), because of the huge number of variables considered in the analysis.

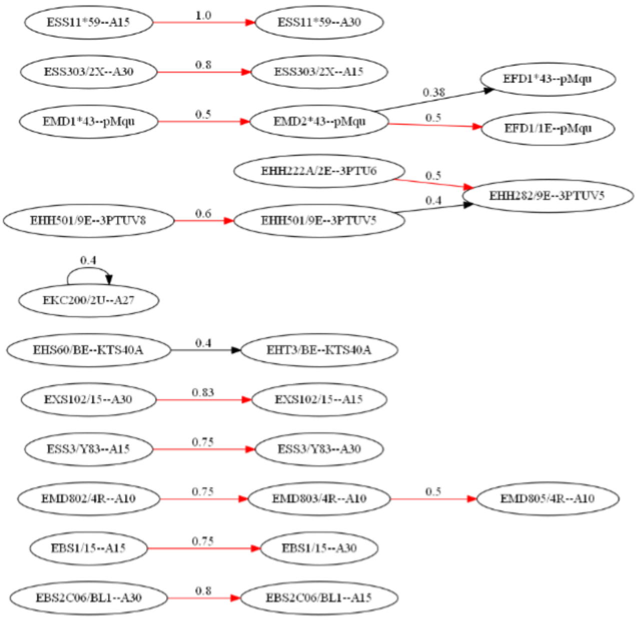


Figure 6.7: An extract of a Markov chain with 150 states, built on the events before a set of given timestamps, with an analysis window of 5 minutes.

6.4 The use of different additional fields

We have seen in Figures 6.5 and 6.6 that by considering the "tag" field we get more specific information that could let us understand better the reasons behind the presence of some frequent transitions (in Markov chains) or strong correlations (in Bayesian networks). A person who knows how these devices work and what the codes in the "tag" field represent can possibly detect the problem, isolate it and solve it in a faster way. So we can say that the networks based on the "tag" field are more useful from the fault detection perspective with respect to the networks based on the device name only. Until now we have presented models based on device names and their "tag" field, but our implementation allows the use of different additional fields.

In Figure 6.8 we show a Bayesian network based on the events after the reference device EMC001*9 with the "level of priority" equal to L3, which is the highest of our discretized priority levels. As we can notice

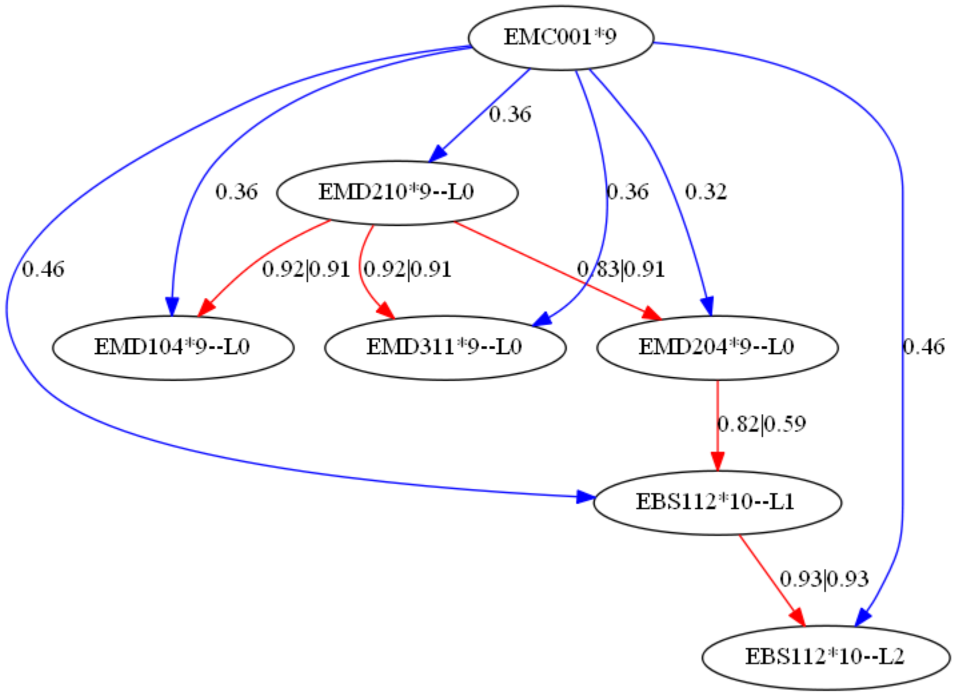


*Figure 6.8: The Bayesian network of reference device EMC001*9 with priority L3, no duplicates, the graphical postprocessing of the reference device and the addition of the field "level of priority".*

in the picture, the fact that we are considering reference events with high priority does not necessarily mean that the events of the devices in the network will have an high priority too. Most of the nodes, in fact, have the lowest priority level (L0), while the nodes related to the same device EBS11*12 has a medium chance (46%) to appear after an L3 event of EMC001*9 with priority L1 or L2. Independently by which of the two priorities we see first, we know (thanks to the inference labels) that we could see the other with a probability of 93%.

We now show different additional fields applied to the same set of

events on which the network in Figure 6.5 is based, and we comment the results obtained from the use of these fields in Markov chain models.
Figure 6.9 is the first variation of the network in Figure 6.5 that we present. It is based on the additional field “description” instead of “tag”. We see that the structure of the network, the name of the devices selected, and the probabilities on the edges have not changed. The only difference is the name of the additional field after the separator “--”. The nodes of this state diagram share the same behaviour of the other corresponding nodes shown in Figures 6.5 and 6.9. This indicates that “description” and “tag” provide information at a similar level of detail, at least in this case. In some of the “description” values we can even see references to other device names, to corresponding tags, or to voltage values. Description values usually give a semantic interpretation to the tag values to which they are related to, and they can help to understand better the scenario of the events happened if we do not know the meaning of “tag” values. It even seems that there exists a one to one correspondence between values of “description” and “tag”, but we have then verified that it is not true in general. Different descriptions can correspond to a single tag and vice versa.
Figure 6.10 shows that the use of the “state” field changes the network structure. The occurrences of the states are higher than those of the tags in Figure 6.5. For this reason and from the names of the states, we can deduce that in this case the information provided by the “state” field is more general with respect to “description” and “tag”. In the example in Figure 6.10, the “state” reports a generic “Fault” in almost all the nodes, grouping together all the different information provided by “tag” and “description”, which explained in more detail the nature of the fault for each device.
In Figure 6.11 we show the Markov chain based on the “level of priority” as additional field. We see that the networks in Figure 6.10 and Figure 6.11 share the same device names and occurrences. In this particular case we have found a perfect correspondence between the “state” and “level of priority” values, but, again, this is not true in general. The

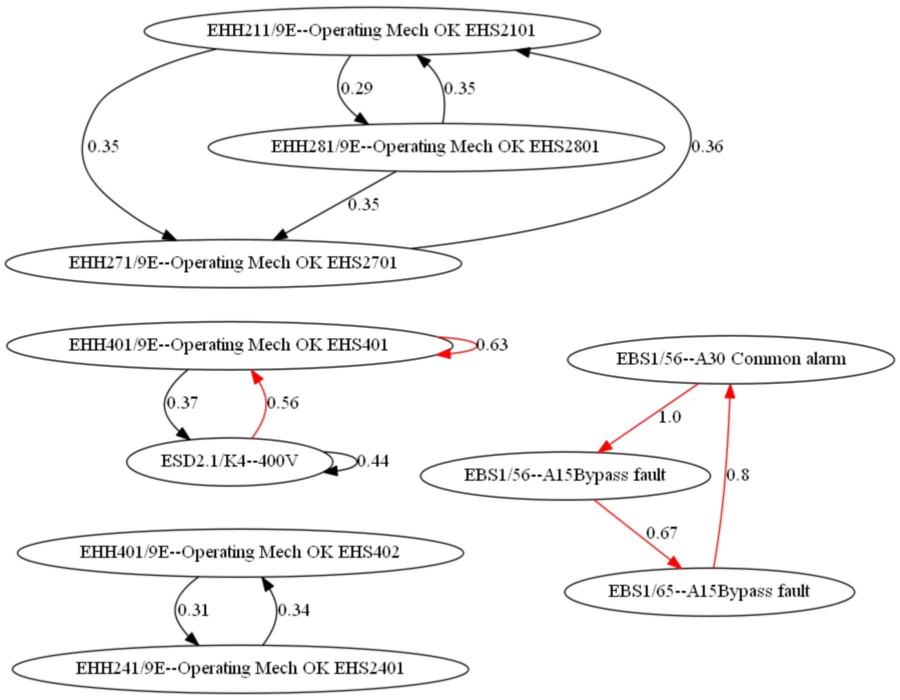

EHH211/9E--Operating Mech OK EHS2101
0.29
0.35
0.35
EHH281/9E--Operating Mech OK EHS2801
0.36
0.35
EHH271/9E--Operating Mech OK EHS2701
EHH401/9E--Operating Mech OK EHS401
0.63
0.37
0.56
ESD2.1/K4--400V
0.44
EBS1/56--A30 Common alarm
1.0
EBS1/56--A15Bypass fault
0.8
0.67
EBS1/65--A15Bypass fault
EHH401/9E--Operating Mech OK EHS402
0.31
0.34
EHH241/9E--Operating Mech OK EHS2401

Figure 6.9: Markov chain related to the device EHS60/BE with priority L1, with states selected with the occurrences" criterion, with "description" as additional field.

possible values of the "level of priority" are only four, while "state" can assume many more values.

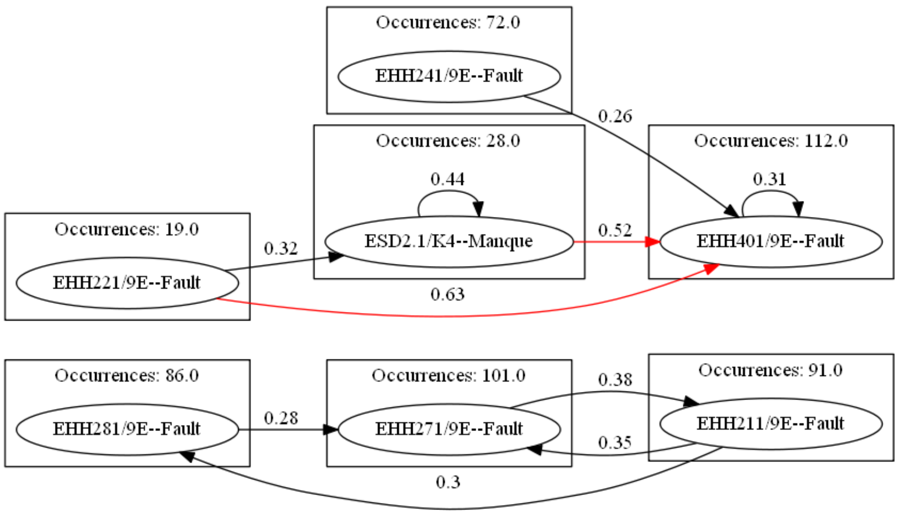


Figure 6.10: Markov chain related to the device EHS60/BE with priority L1, with states selected with the close pairs criterion and "state" as additional field.

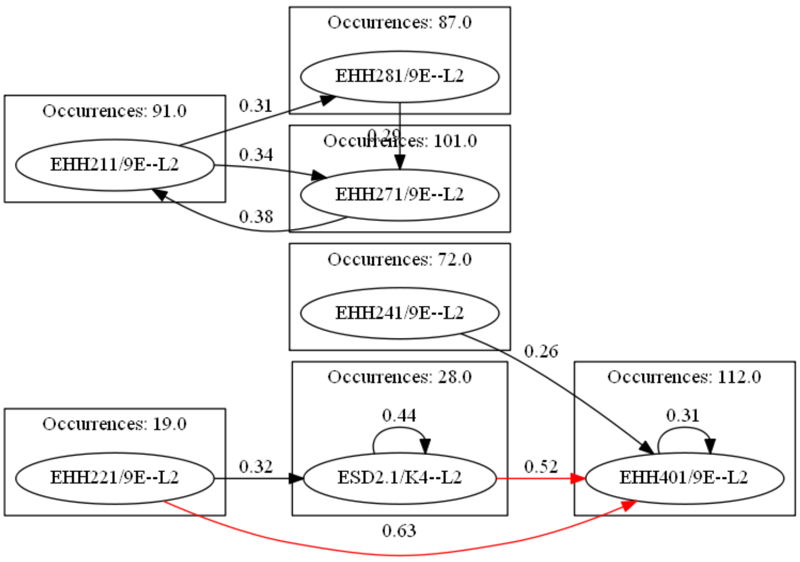


Figure 6.11: Markov chain related to the device EHS60/BE with priority L1, with states selected with the close pairs criterion and "level of priority" as additional field.

6.5 The window of analysis

Often in this document we assumed a value of W equal to 5 minutes. However, in general, there might not be a perfect length for the window. An analysis can be repeated many times with multiple W values and lead to different valid conclusions. For example, there might be some devices that are influenced by the anomaly happened at the timestamps T only after an interval greater than 5 minutes; others might regularly appear within 1 minute but they might not be chosen by the variable selection method if the window is too large. Clearly, the effectiveness of the size of the window is also bound to the preprocessing and variable selection technique chosen. For example, if our goal is to use temporal criteria to find devices that appear immediately after (or before) the timestamps, we should keep W as small as possible to reduce the noise (as mentioned in Section 4.2.1.3). If we want to find devices with the highest number of occurrences, we can keep the window relatively large since the variable ranking should not be excessively affected by the introduction of noise. However, we should also be particularly careful when enlarging the window because timestamps close to each other may start to overlap, leading to more duplications of events.

Regarding the timestamps extracted from reference devices we noticed that the distribution of data follows some regular patterns. After a timestamp, most of the events are reported in the very first minutes. With W of one or two minutes, ranking of variables may change significantly. Then, the more we increase the window, the more are the chances that the number of events will start to stabilize. An image with the trends of some of the devices utilized for our analysis is shown in Figure 6.12. The graph shows on the vertical axis the number of events reported after the timestamps of a reference device with priority L1, while on the horizontal axis there is the size of the window W. Each line is related to a single reference device. The number of events are calculated for every device exactly every minute, starting from $W = 1$. As we can see in the graph, a good amount of events are already reported in the first minute. Most of the spikes tends to

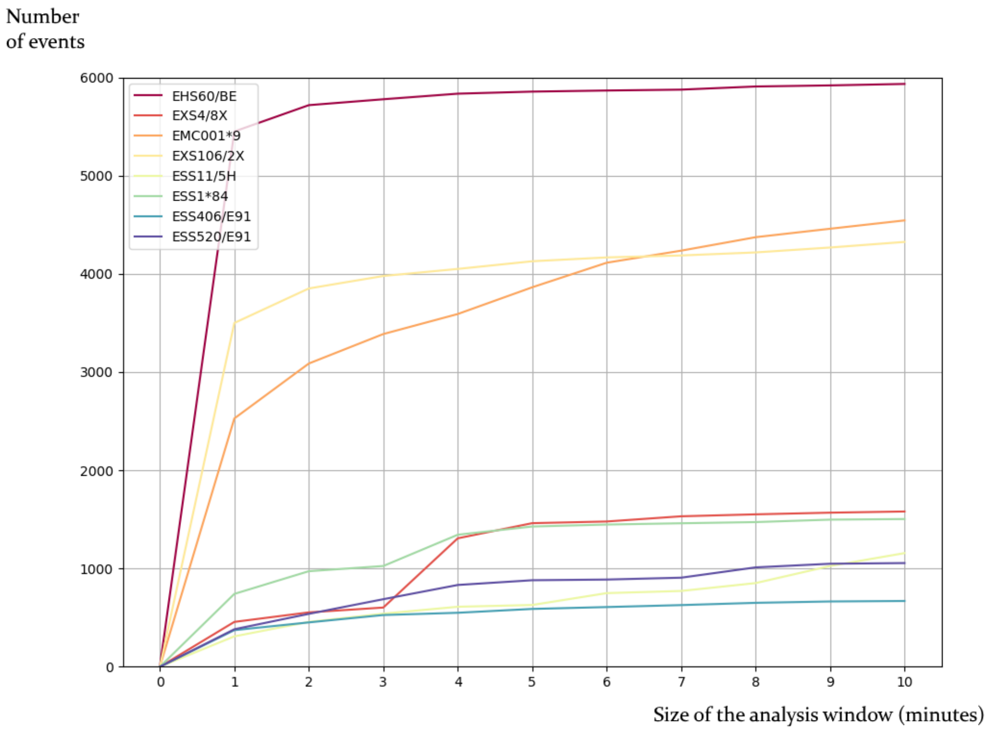


Figure 6.12: A graph that puts in relation the number of events (after the timestamps) with the dimension of the analysis window. This graph considers only events of reference devices with priority L1.

happen before 5 minutes. One of these spikes, for example the one of device EXS4/8X, happens between 3 and 4 minutes. For these devices, setting increasingly larger values of W over 5 minutes should not provide much different models. However, setting a value of W larger than 5 minutes does not seem a good idea by looking at the graph, since it is more probable to include noise in the analysis than to consider new interesting data. On the other hand, if we choose a low value of W, we risk to leave out possible significant correlations. The choice of $W = 5$ is a good compromise in most cases because we reduce to the minimum the possibility of ignoring relevant data, while managing to keep a rather low the value of W.

The graph for events of reference devices with priority L2 is shown in Figure 6.13. The graphs based on priority L1 and L2 of the ref-

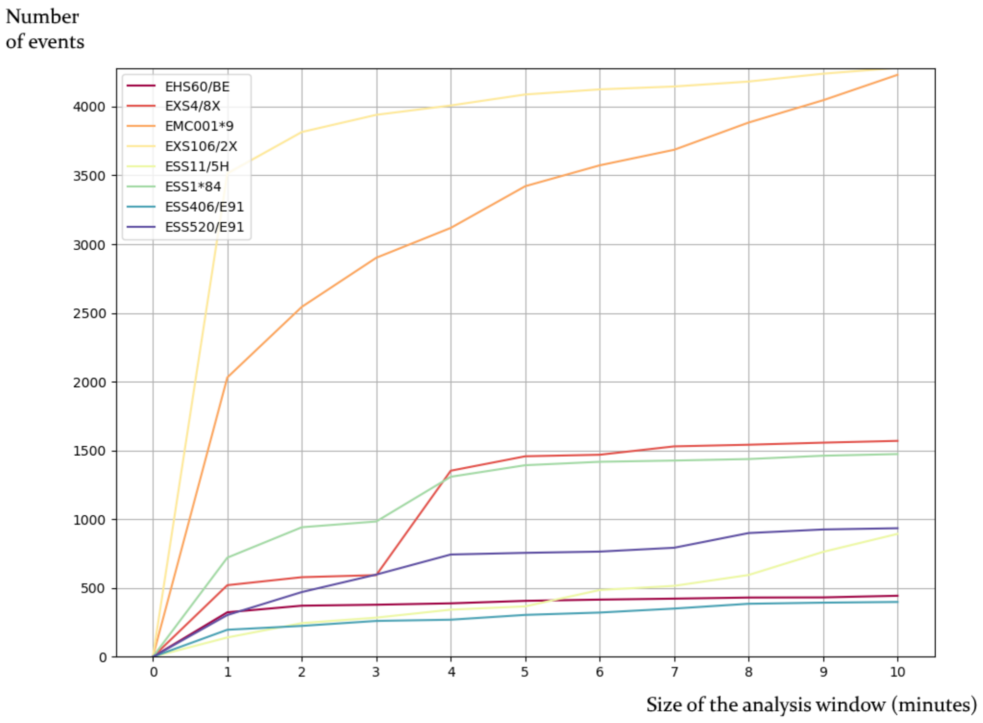


Figure 6.13: A graph that puts in relation the number of events (after the timestamps) with the dimension of the analysis window. This graph considers only events of reference devices with priority L2.

erence devices present similar trends for most of the devices, apart for EHS60/BE. We can even spot the same interesting spike of the EXS4/8X trend between $W = 3$ and $W = 4$. Even in this case, $W = 5$ seems a good choice considering the general trend of all devices. However for some specific devices, other values of W may be more appropriated. We see that some devices stabilize their trends sooner than 5 minutes, while the trends of EMC001*9, and in minor part of EXS106/2X and ESS11/5H, continue to grow at a non negligible rate even after 9 minutes.

The situation before the timestamps extracted from a reference device is similar. Figure 6.14 shows the graph for the number of events before some reference devices with events of priority L1. The more the analysis window size increases, the more we move further in the past. At the value of minutes equal to 10 we are actually looking at a correlation window of $W = -10$ minutes. As we can see, for W

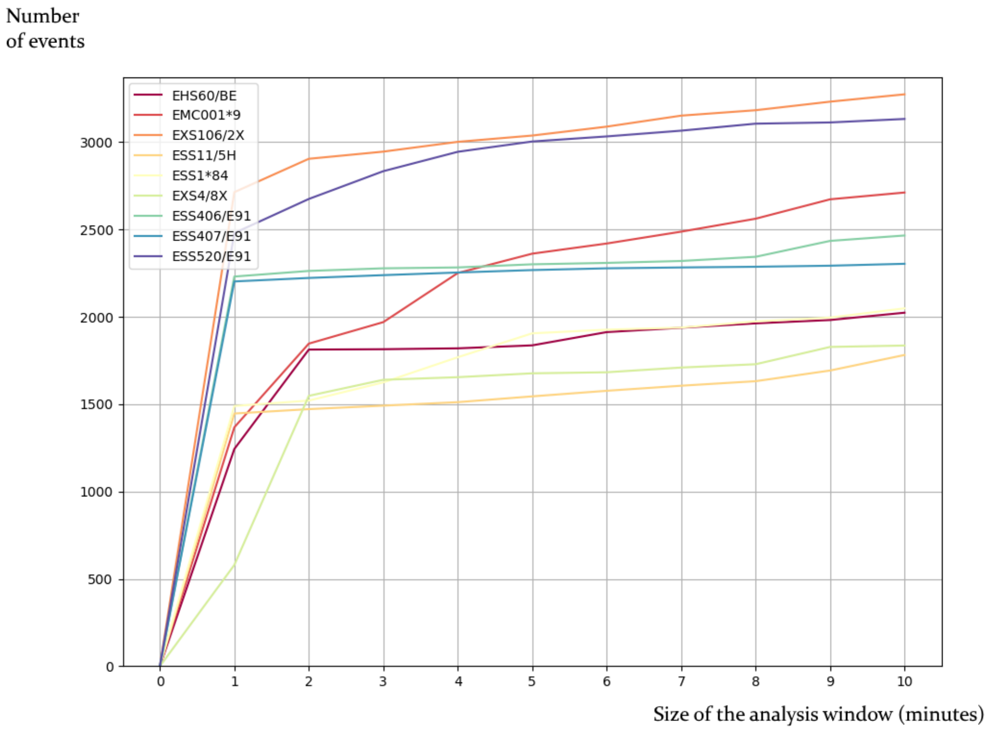


Figure 6.14: A graph that puts in relation the number of events (before the timestamps) with the dimension of the analysis window. This graph considers only events of reference devices with priority L1.

around 5 minutes the number of events does not increase very much anymore. Is not surprising that the highest number of events reported is between 0 and -1 minutes, since we can suppose that immediately before an important event of a reference device other events could have been reported, maybe related to the same fault that caused the event of the reference device in the first place.

6.6 A comparison between models

As we discussed in Section 5.1.2, Markov chains could be used to discover the temporal order of activation of some variables that were found to be correlated in a Bayesian network. In Figure 6.15 we show a direct comparison between the two models, both generated on the same set V of devices related to the reference device EXS106/2X with priority L1. First of all, we notice that the general structure of the two networks is similar. EBS1/22 acts as a point of separation between EBS132/2X and the rest of the network. The devices that form a sequence of connections between EBS1/22 and ESS10/1DX in the Bayesian network are present also in the Markov chain, but there are transitions that break the sequence by connecting them directly to EBS11/22. Moreover, the order of the devices in the sequence is not the same.
This is not a contradiction with respect to the Bayesian model results. In fact, since the sequence in the Bayesian network is composed of quite high inference label values, it is perfectly normal that a parent node (like EBS1/22) may have a strong direct correlation (of the type 1-1) with a node that is not a direct child (like ESS10/1DX), which explains why the same connections are present in the Markov chain and why some nodes are swapped.
Moreover, this is also one of the main reasons why we do not see high values of transition probabilities between devices that have high correlations in the Bayesian network. The model on the left of Figure 6.15, in fact, does not take the order of devices into consideration. While looking at the Markov chain, instead, we discover such order and we realize that devices between EBS1/22 and ESS10/1DX (in the Bayesian network) do not always appear in that precise order, otherwise we

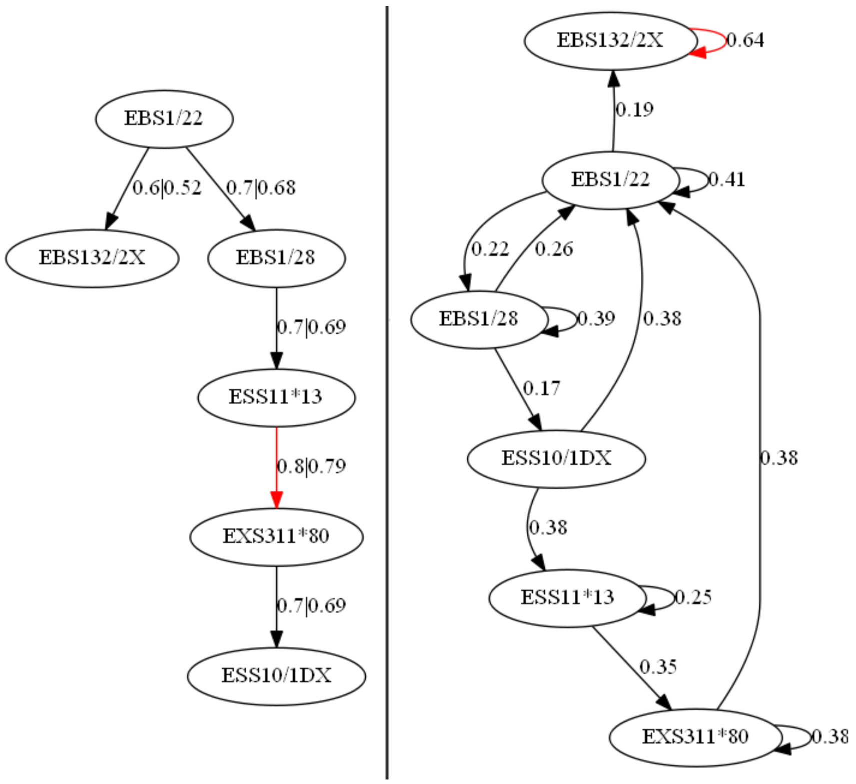


Figure 6.15: Two models related to the reference device EXS106/2X, with priority L1. On the left, the Bayesian network (generated without duplicates). On the right, the Markov chain (with visualization threshold set to 0.15).

would have seen a chain of devices having transitions with probabilities equal to 1. Sometimes, the same device reports more events (for example the device EBS1/28, that has a self-loop with probability 0.39) or the chain restarts from EBS1/22 (see for example the transition between EBS1/28 and EBS1/22, or the transition between EXS311*80 and EBS1/22). All these factors cause the probability of transitions between devices in the sequence to decrease and be much lower than the inference labels.

In the previous discussion we often talked about a sequence of devices, that is meaningful only in the Markov chain model. However, we do not know from which device this chain can actually start. With the addition of the reference device (in Figure 6.16) we discover this information. We see that EBS1/22, which seemed a device that could

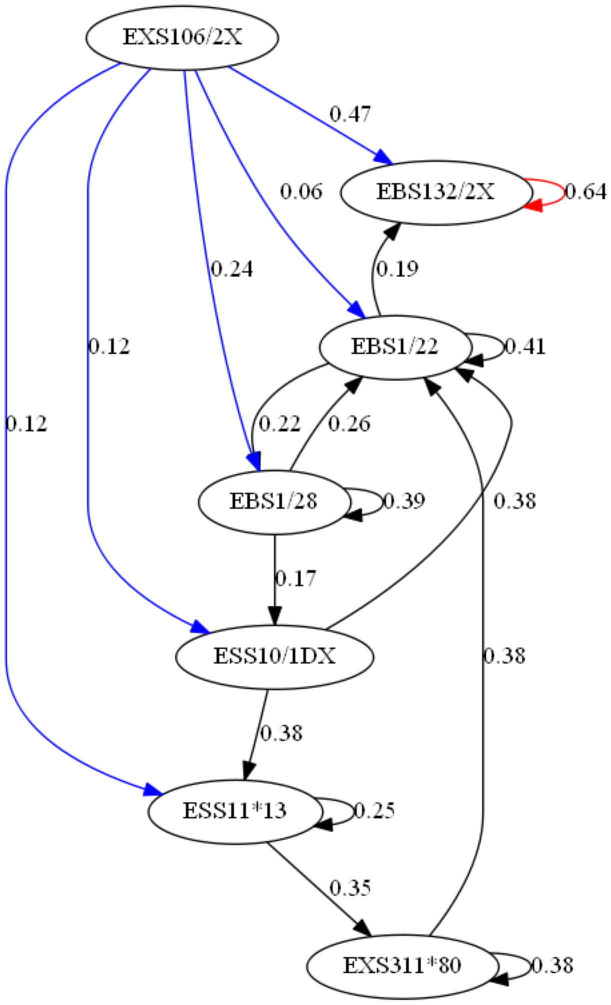


Figure 6.16: The Markov chain on the right of Figure 6.15 with the addition of the reference device.

have started the chain, is actually almost never (6%) the first after an event of EXS106/2X with priority L1. Instead, we see that almost half of the times EBS132/2X appears as first, while EBS1/28 is the second most likely (24%) device, which is also part of the chain between EBS1/22 and EXS311*80. We also notice that EXS311*80 never appears as first device after EXS106/2X, which means that its activation could be caused by the devices in the network (like ESS11*13), and not by EXS106/2X.

6.7 Networks with multiple reference device

In Section 4.1.1.1 we have introduced the possibility of extracting timestamps from multiple reference devices, that also become the devices in the set V that we see in our network. This technique is useful when we want to focus the analysis on the relationship between a set of devices, without any dependence from a single reference device or arbitrary chosen timestamps. Consider the Bayesian network of the reference device EHS60/BE generated with the frequency criterion and priority L1, shown in Figure 6.17. We notice that all devices in the network appear after an event of priority L1 of EHS60/BE with probabilities that range from 33% to 41%. We also see that the correlations between nodes are very high. At this point we may wonder if such correlations are valid only after EHS60/BE or they actually are independent from the reference device. To find out this, we select the five variables of the Bayesian network as the multiple reference devices, from which we extract the timestamp set T. Since we want to focus on these devices only, we manually select them to compose the set V. We consider events of all priority levels. Figure 6.18 shows the result of this procedure. We notice that the network does not present any valid correlation. So, it seems that the correlations between these devices are not valid in general. The values of the inference labels that we see in the network are really low, but not always equal to zero. This is not a surprise, since we expect that the correlations found in Figure 6.17

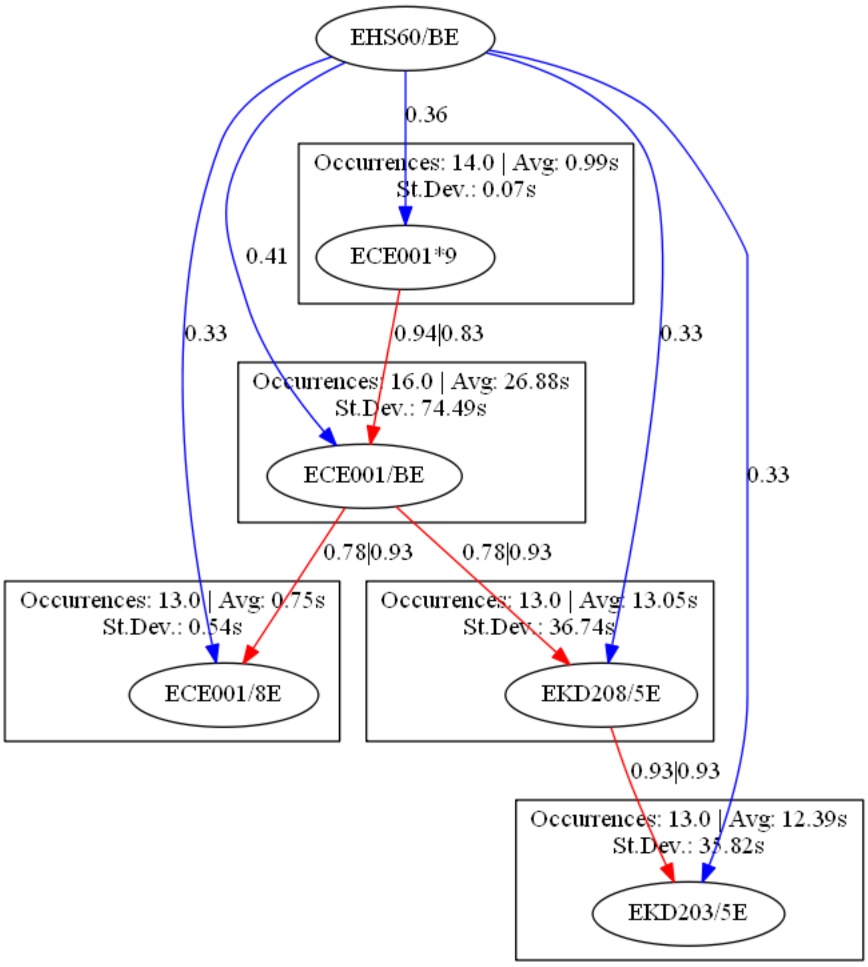


Figure 6.17: The Bayesian network of the reference device EHS60/BE, generated without duplicates, priority L1 and the frequency criterion.

should be found also with the multiple reference devices setting. In the network based on multiple reference devices, however, the number of occurrences of events that are not correlated is much higher than the occurrences of correlated devices found also in the original network. The difference in the number of occurrences explains the absence of

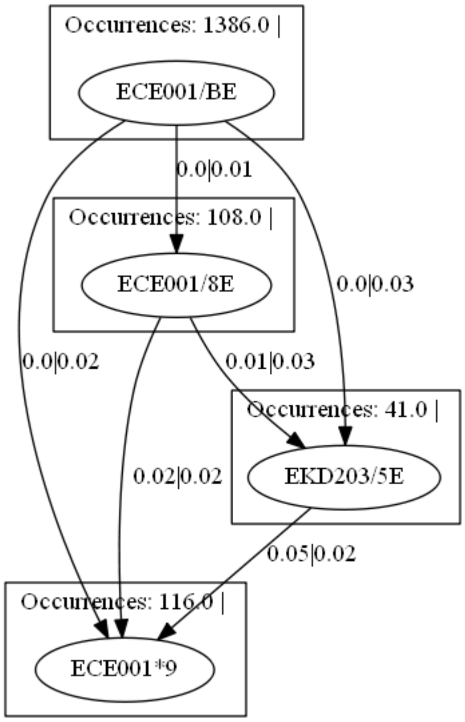


Figure 6.18: The Bayesian network generated from the multiple reference devices in Figure 6.17, generated with no duplicates.

relevant correlations in Figure 6.18. If the networks in Figure 6.17 and in Figure 6.18 were similar, even if based on a significantly different amount of occurrences for the same devices, we would have concluded that the presence of the reference device would not have been relevant to describe the behaviour of the other variables of the network. Great differences between those networks, instead, prove that the results in Figure 6.17 are useful to describe what happens after the events of “EHS60/BE” with priority L1, and that they don’t represent general trends in the data that are also valid when “EHS60/BE” does not register events in the log.

Chapter 7

Conclusions and future development

In this chapter we wrap up the main conclusions of our work (in Section 7.1) and propose some improvements that could enhance the effectiveness of our analysis (in Section 7.2).

7.1 Conclusions

In this work we have shown how Bayesian networks and Markov chains can be applied in a data-driven context to analyze and predict the effect of anomalies. Specifically, we have studied the relationship between the components of a complex electrical infrastructure, starting from the events generated directly by these components. Differently from most of the approaches to this topic, our models are not built on the nominal behaviour of the system but on the anomalous events that are reported on the system's log. The first and simplest model we have introduced for this purpose is a Bayesian network with variables based on the device names. Through our discussion in Section 4.2, we realized that Bayesian networks can be useful to detect correlations between devices but are not appropriate to reason about the temporal causality of events. With the presentation of preprocessing techniques

and variable selection criteria, we have shown how the results of this model can be generated to fit the focus of our analysis. After testing different configurations we have drawn some conclusions. For example, we found out that the networks based on tags and device names are generally the most useful, in that they allow to gain much more specific insights on the events occurred to the devices. The temporal variable selection criteria, instead, is a good way to introduce temporal constraints on the variables we show in the network.
However, to expand our tools and really take into account time and sequentiality we introduced a second model, the Markov chains. This new model naturally deals with the time dimension and allowed us to extract information complementary to the ones found with Bayesian networks, which can help to identify possible causes and consequences in the correlations learned with Bayesian networks based on the same devices and set of timestamps. The presentation of these two models followed a parallel structure, sharing many preprocessing, variable selection, and postprocessing techniques, making it easier to compare them.

A great effort has been put to increase the quality of the data used for the model learning, in order to create networks whose probabilities were closest as possible to the real values. This explains the focus on the preprocessing techniques like duplicate elimination for Bayesian networks, device filtering for Markov chains, and clustering for both models. The clarity of visualization of the generated models is another recurrent theme of this work. Several post-processing options have been developed in order to facilitate the extraction of the most relevant information in a short time. Clear visualization capabilities, however, come with a cost: we cannot select all the devices that *could* be correlated, which is the reason why we had to develop criteria to choose the most relevant subset of variables to analyze.

7.2 Future development

Many of the ideas applied in this work can be developed even further. We have seen the benefits obtained with a more specific representation of variables based on the name of the device and additional fields like "tag" or "description". We could extend this representation also to the reference device, which in our analysis was constrained to be a device with events of a particular priority level. The choice of a reference device based on tags instead of the level of priority seems a promising approach. With this modification, criteria based on temporal distance and confidence with respect to timestamps can hopefully find even more precise patterns.

The close pairs criterion for Markov chains showed a first approach based on the adjacency of variables in sequences, which is at the basis of what determines the usefulness of the model generated. For this reason, we believe it is worth to try different approaches following this direction. We have seen in Section 6.3 that the actual version of the close pairs criterion is biased on the selection of devices with an high number of occurrences. The ranking of the variables could be modified applying the reasoning done for the confidence criterion described in Section 4.2.1.4 to the adjacent elements in Markov chains' training sequences. The close pairs criterion ranks the pairs on the basis of the number of times they appear as consecutive in the training sequences. If that number is divided by the number of occurrences of the first element of the pair, we would create a close pairs criterion based on confidence. It would search directly for all the highest values of transition probabilities and would select the devices which are sources or destinations of the edges with the highest probability. A threshold value in order to consider only variables with a minimum number of occurrences is needed to avoid a bias for less frequent variables, for the same reasons explained for the confidence criterion in Section 4.2.1.4. A temporal approach for the selection of variables for Markov chains based on the minimum average time of appearance between adjacent devices in the training sequences can be tried too. Average, standard deviation of time intervals, or the sum of both can be chosen as criteria with the same advantages and disadvantages shown in Section 4.2.1.3 for Bayesian networks and the relation with the timestamps i. We can call this criterion *temporal close pairs*.

Another promising direction of research for this problem regards the change of the Boolean values currently used in the domain of variables in Bayesian networks. Instead of embedding an additional field in the variable name, as done in this work, it could be useful to represent the nodes of the network always with the device names, and use the values assumed by their additional fields as domains of the variables. For example, a variable could assume 4 values that correspond to its four different tags reported in the log. Many advantages come with this approach: it becomes possible to represent all the pairs of device names and additional fields for every device considered for the analysis; encoding the same amount of information in a Bayesian network with a Boolean representation of values would take a much larger number of nodes. The structure of the Bayesian networks learned could change completely and the independence tests between the same devices could give very different results. With this new formalization of the problem, the complexity would be transferred to the CPD tables. A possible disadvantage could be the increase of network complexity. The time to learn the network structure may also increase when adding more values in the domains, but at the same time we could be able to represent more information with less nodes, so further research is needed to understand the performance loss or gain with respect to our current representation.
However, it would be impossible to use inference labels for visualization, since a lot of interesting probabilities could be found inside a single CPD, and inference queries would become the only way to get information from the model. Another problem is related to the number of samples for the pairs of devices and additional fields like tag or description, as shown in Section 6.4. A way to cope with both problems of the CPD complexity and data samples is, assuming to consider

tags as additional field values, to use the most frequent tags as values for the variables, and to group all the other possible values with few examples in a single value called "Other values".

In this work, the analysis of the results of our models have been done manually. Correlations and inference must be performed on Bayesian networks, while sequentiality and temporality must be observed in Markov chains. An operator who wants to use these models must have a minimum level of knowledge on how these techniques work, or otherwise he/she might misinterpret the results. A possible improvement in this sense is to create a framework that automatically derives the consequences of an anomaly as soon as it happens, and that integrates the results from our models (and possibly additional data analysis) to give the operator information that do not depend by the displayability and readability of the models.

Bibliography

[1] Machine learning and data mining by Pier Luca Lanzi. `http://www.lourenco.ws/pier-luca-lanzi-follow-machine-learning-and-data-mining/13-nearest-neighbor-and-bayesian-classifiers`. Accessed: 11-03-2018.

[2] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

[3] Anders Tolver. An introduction to Markov chains. 2016. Department of Mathematical Sciences, University of Copenhagen.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[5] Maurilio P. Coutinho, Germano Lambert-Torres, L.E.Borges da Silva, HG. Martins, Horst Lazarek, and J. Cabral Neto. Anomaly detection in power system control center critical infrastructures using rough classification algorithm. In *Proceedings of the 3rd IEEE International Conference on Digital Ecosystems and Technologies*, DEST'09, pages 733–738. IEEE, 2009.

[6] Marco Martinelli, Enrico Tronci, Giovanni Dipoppa, and Claudio Balducelli. Electric power system anomaly detection using neural networks. In *Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 1242–1248. Springer, 2004.

[7] Adnan Anwar and Abdun Naser Mahmood. Anomaly detection in

electric network database of smart grid: graph matching approach. *Electric Power Systems Research*, 133:51–62, 2016.

[8] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1285–1298, New York, NY, USA, 2017.

[9] Kevin Murphy. A brief introduction to graphical models and Bayesian networks. `https://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html#appl`, 1998. Accessed: 11-03-2018.

[10] Zhiqiang Cai, Shudong Sun, Shubin Si, and Ning Wang. Research of failure prediction Bayesian network model. In *Proceedings of the 2009 16th International Conference on Industrial Engineering and Engineering Management*, pages 2021–2025, Oct 2009.

[11] Dejan P. Jovanović and Philip K. Pollett. *Distributed Fault Detection Using Consensus of Markov Chains*, pages 85–105. Springer Netherlands, Dordrecht, 2014.

[12] Abida Haque, Alexandra DeLucia, and Elisabeth Baseman. Markov chain modeling for anomaly detection in high performance computing system logs, 11 2017.

[13] Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers Chemical Engineering*, 27(3):293 – 311, 2003.

[14] Venkat Venkatasubramanian, Raghunathan Rengaswamy, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies. *Computers Chemical Engineering*, 27(3):313 – 326, 2003.

[15] Hongyu Wang, Zuohua Tian, Songjiao Shi, and Zhenxin Weng. Fault detection and isolation scheme based on parity space method for discrete time-delay system. In Wei Zhang, editor, *Fault Detection*, chapter 5. InTech, 2010.

[16] Yiannis Papadopoulos. Model-based system monitoring and diagnosis of failures using statecharts and fault trees. *Reliability Engineering System Safety*, 81(3):325 – 341, 2003.

[17] William J. Puglia and Bahman Atefi. Examination of issues related to the development and implementation of real-time operational safety monitoring tools in the nuclear power industry. *Reliability Engineering System Safety*, 49(2):189 – 199, 1995.

[18] Claire Palmer and Paul W.H. Chung. Verifying signed directed graph models for process plants. *Computers Chemical Engineering*, 23:S391 – S394, 1999.

[19] Lioun Wee Chen and Mohammad Modarres. Hierarchical decision process for fault administration. *Computers Chemical Engineering*, 16(5):425 – 448, 1992.

[20] Ole J. Mengshoel, Mark Chavira, Keith Cascio, Scott Poll, Adnan Darwiche, and Serdar Uckun. Probabilistic model-based diagnosis: An electrical power system case study. *Proceedings of the IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 40(5):874–885, Sept 2010.

[21] Tarem Ahmed, Boris Oreshkin, and Mark Coates. Machine learning approaches to network anomaly detection. In *Proceedings of the 2Nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, SYSML'07, pages 7:1–7:6, Berkeley, CA, USA, 2007.

[22] Wil van der Aalst. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems*, 3(2):7:1–7:17, July 2012.

[23] Wil van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar,

Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny S. Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid R. Motahari-Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn. Process mining manifesto. In Florian Daniel, Kamel Barkaoui, and Schahram Dustdar, editors, *Business Process Management Workshops*, pages 169–194, Berlin, Heidelberg, 2012. Springer.

[24] Fábio Bezerra, Jacques Wainer, and W. M. P. van der Aalst. Anomaly detection using process mining. In Terry Halpin, John Krogstie, Selmin Nurcan, Erik Proper, Rainer Schmidt, Pnina Soffer, and Roland Ukor, editors, *Enterprise, Business-Process and Information Systems Modeling*, pages 149–161, Berlin, Heidelberg, 2009. Springer.

[25] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, Henricus M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The prom framework: A new era in process mining tool support. In Gianfranco Ciardo and Philippe Darondeau, editors, *Applications and Theory of Petri Nets 2005*, pages 444–454, Berlin, Heidelberg, 2005. Springer.

[26] Luis M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, (7):2149–2187, 2006.

[27] Ron Kenett. Applications of Bayesian networks. 10 2012. Neaman Institute for National Policy Research, the Technion; Institute for Drug Research, School of Pharmacy, Hebrew University.

[28] Philippe Weber, Gabriela Medina-Oliva, Christophe. Simon, and Benoit Iung. Overview on Bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25(4):671–682, June 2012.

[29] Brett Drury, Jorge Valverde-Rebaza, Maria-Fernanda Moura, and Alneu de Andrade Lopes. A survey of the applications of Bayesian networks in agriculture. *Engineering Applications of Artificial Intelligence*, 65:29 – 42, 2017.

[30] Baoping Cai, Lei Huang, and Min Xie. Bayesian networks in fault diagnosis. *IEEE Transactions on Industrial Informatics*, 13(5):2227–2240, Oct 2017.

[31] Jinqiu Hu, Laibin Zhang, Zhansheng Cai, Yu Wang, and Anqi Wang. Fault propagation behavior study and root cause reasoning with dynamic Bayesian network based framework. *Process Safety and Environmental Protection*, 97:25 – 36, 2015.

[32] M. Nyberg. Failure propagation modeling for safety analysis using causal bayesian networks. In *2013 Conference on Control and Fault-Tolerant Systems (SysTol)*, pages 91–97, Oct 2013.

[33] Philipp Von Hilgers and Amy N Langville. The five greatest applications of markov chains. In *Proceedings of the Markov Anniversary Meeting, Boston Press, Boston, MA*. Citeseer, 2006.

[34] Prerna Rai and Arvind Lal. Google pagerank algorithm: Markov chain model and hidden markov model. *International Journal of Computer Applications*, 138(9):9–13, 2016.

[35] Nong Ye. A markov chain model of temporal behavior for anomaly detection. In *In Proceedings of the 2000 IEEE Workshop on Information Assurance and Security*, pages 171–174, 2000.

[36] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *IEEE*

Transactions on Knowledge and Data Engineering, 16(9):1128–1142, 2004.

[37] Dion H. Goh and Rebecca P. Ang. An introduction to association rule mining: An application in counseling and help-seeking behavior of adolescents. *Behavior Research Methods*, 39(2):259–266, May 2007.

[38] Classification rules - slide by professor Pier Luca Lanzi. `https://www.slideshare.net/pierluca.lanzi/machine-learning-and-data-mining-12-classification-rules`. Accessed: 18-03-2018.

[39] Wei Shen, Jianyong Wang, and Jiawei Han. *Sequential Pattern Mining*, pages 261–282. Springer International Publishing, Cham, 2014.

[40] Pgmpy 0.1.2 documentation. `http://pgmpy.org/`. Accessed: 18-03-2018.

[41] Pomegranate's documentation. `http://pomegranate.readthedocs.io/en/latest/MarkovChain.html`. Accessed: 18-03-2018.

[42] Graphviz's official website. `http://www.graphviz.org/`. Accessed: 18-03-2018.

[43] Scikit-learn's official website. `http://scikit-learn.org`. Accessed: 18-03-2018.

[44] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[45] Mean shift clustering. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf`. Accessed: 20-03-2018.

[46] Dbscan: Density-based spatial clustering of applications with noise. `http://www.cs.fsu.edu/~ackerman/CIS5930/notes/DBSCAN.pdf`. Accessed: 19-03-2018.