Diagnostic under Uncertainty

Patrick Rammelt, Peter Tondl, Ulrich Siebel
Carmeq GmbH
[Patrick.Rammelt | Peter.Tondl | Ulrich.Siebel]@carmeq.com

Abstract

This paper introduces a method for making probabilistic diagnostics in complex technical systems, e.g. known from the automotive and the avionic sector. Using probabilities to account for uncertainties provides a natural way for sequencing suspected components, such that malfunctions could be eliminated in a minimum of time or with a minimal loss of money due to avoidance of unnecessary actions. In the approach presented here a probabilistic system is modeled by Bayesian networks.

1 Introduction

Diagnostics in technical systems mean to identify and finally repair malfunctioning components within a minimum of time and/or with the minimal loss of money.

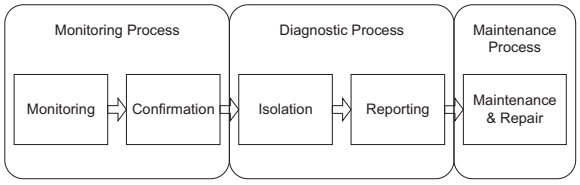

Figure 1: Monitoring, Diagnostics and Maintenance Task Chain [1]

Figure 1 shows the process chain for a diagnostic system from *monitoring* (observing possible problems) over *confirmation* (e.g. by performing a self test), *isolation* (see below) and *reporting* to the final step called *maintenance* ("repairing") - see [1] for a more detailed description. In this paper we will focus ourselves to the *isolation task*, which is shown in more detail in fig 2 . "Isolation" means to use all available information in order to narrow the possible causes of a problem down to those which are in accordance with the available information. This task itself consists of several components: *data aquisition* means to get all data needed; from this a *state detection* generates *indicators* used by the *diagnostic engine* whose results could finally undergo some *post-processing*. Here, we mainly describe an approach for a *diagnostic engine*, also having a short look at an expedient *result post-processing* technique.

Diagnostics in technical systems are still often based on a "strict" kind of logic. Some systems explicitly use Boolean logic and the commonly known logical rules in order to detect faults - see [2] for an example. Other approaches are implicitly based on Boolean assumptions, in the sense that they do not include any certainty factors provided with the list of possibly faulty components (e.g. in [3]).

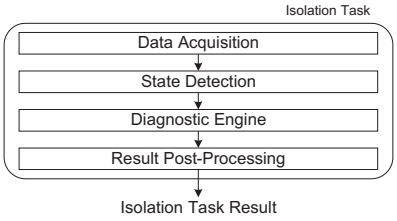

Figure 2: Isolation Task [1]

There are also other approaches using fuzzy logic or (Bayesian) probabilities (e.g. in [4]). The main advantage is that such approaches are able to sort the resulting list of possibly faulty components by the fault-probability of each component[1].

One commonly used method to create consistent probability models are Bayesian networks. The systems addressed in this paper include typically a huge number of components and provide large sets of indicators which are sensitive to several possible malfunctions of these systems. One challenge of using Bayesian networks for such systems is to find a structure which results in a Bayesian network of manageable compact size (see also [5] and [6]). Even if such a representation has been found it is still another challenge to propagate the incoming information about observed indicators through the network

[1] or even better by the expected loss for checking each component, as described later in this paper

and calculate the resulting posterior fault probabilities for all components in an efficient way. We will address both challenges in this paper: finding a compact structure and propagating information through the resulting Bayesian network.

2 Basic Principles

We will give basic definitions and basic principles how Bayesian networks for diagnostic systems can be created and which commonly known calculation schemes are applicable for those networks.

2.1 Cause and Effect

This section gives a short overview about definitions related to the probability-based diagnostic approach.

Definition 1 (Cause)
We will use the terms "cause", "module" and sometimes "component" in a similar sense, where "cause" is the most general circumscription for something that potentially becomes subject to diagnostics, i.e. something that could cause any trouble. □

Definition 2 (Effect)
Likewise we use the terms "effect" and "indicator" both for the observable effects used to distinguish a system that runs o.k. from one that is suffering from any malfunctions. Trouble-codes which are readable from the controller units of a system could be used for that, but also any other (potentially) observable effect could be an indicator. □

Definition 3 (Engine)
With the terms "(diagnostic) engine" or "system" for short, we denote the whole Bayesian network[2] and the procedures used to propagate information through such a network. □

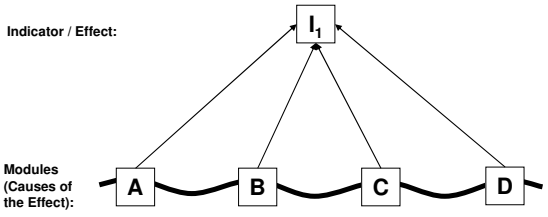

Figure 3: Entities of a diagnostic system

[2]Apart from other nodes, such a network has one node for each cause and each effect.

2.2 Causal Diagnostics

First, a system is described by a set of modules. Depending on the system, modules can be hardware components as well as software modules or anything else which might suffer from malfunctioning[3].

Second, the system is defined by a set of indicators, which show the state of (parts of) the system. Indicators are binary, i.e. saying "No problems" or "Something's wrong here".

An indicator could be set by several causes, i.e. the indicator cannot distinguish whether the problem is due to a malfunction of module A or of module B. Vice versa we can say that all modules connected to an indicator must be o.k. if the indicator doesn't report any problem (not yet regarding any uncertainties). The set of modules effecting an indicator I_1 and the set effecting an indicator I_2 could overlap[4]. In order to isolate single causes it is in fact essential that the sets of causes do overlap.

In our approach we use three types of probabilities:

- First of all, every cause has a **prior probability** for each possible state (e.g. "faulty" and "healthy"). The sum over all prior probabilities of one cause is 1.
- Second, for each effect there is one probability that the indicator is set (indicating a problem) although none of the causes justifies the setting (none of them is faulty). We call this probability "**false positive probability**".
- Third, for each connection between a cause and an effect there is a "**sensitivity**" given, i.e. the probability that the indicator is set given the underlying component is in fact faulty. If the cause has more than one fault-state representing different fault-types, for each of them a sensitivity has to be defined[5].

2.3 Network Structure

The structure shown in fig 3 could constitute a Bayesian network. The problem is that the probability table which has to be defined for each indicator-node grows exponentially with the number of parents (causes) in such a Bayesian network[6].

[3]Module types could also be mixed.

[4]Two indicators can even have the very same set of connected modules.

[5]It is also possible to define a sensitivity for the indicator to be set given the causing module is healthy, but in this paper we assume that value to be 0 for all cause-effect-connections.

[6]In a Bayesian network, a conditional probability table (CPT) has to be provided for each node. The CPT de-

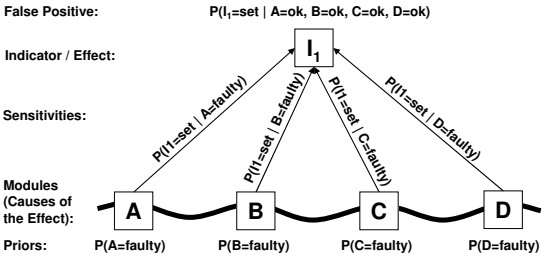

Figure 4: Three types of probabilities

To avoid this problem we introduce two types of intermediate nodes:

- First, for every connection of a cause to an effect node there is one **sensor node** added.
- Second, multiple levels of **intermediate combinator nodes** (ICN) are used. Each node combines two nodes from the previous level. For an effect-node having one or two causes no such node is needed. For three causes one ICN is needed, for four causes two ICNs on the same level are needed. For 8 causes 4 ICNs on the first and two ICNs on the second level are needed - and so on.

As shown in fig 5, in this structure we can easily introduce the three types of probabilities mentioned in section 2.2[7].

The structure could be either created from experts knowledge or from a wiring diagram or other technical descriptions. Likewise, the probabilities could be given by experts or taken from known failure rates, reliabilities, etc.

3 Propagation

The advantage of using bayesian networks is, that it is not necessary to observe all indicators at a time. There might be circumstances where only a subset of all observable indicators are in fact available. On the other hand, observations are not limited to the indicator-nodes. It is possible to include direct knowledge about some causes as well. For example if in a first run of the diagnostic system a component A has been detected to be the most probable

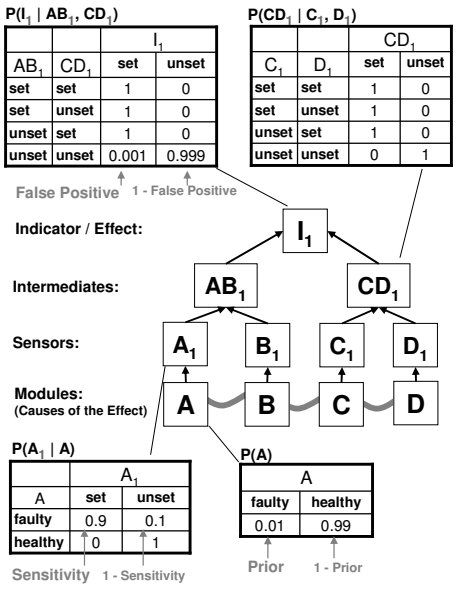

Figure 5: Conditional probability tables for a diagnostic type Bayesian network

reason for a detected malfunction, but that component was checked and found to be healthy, then this observation could be fed into the bayesian network and is used for a second run[8].

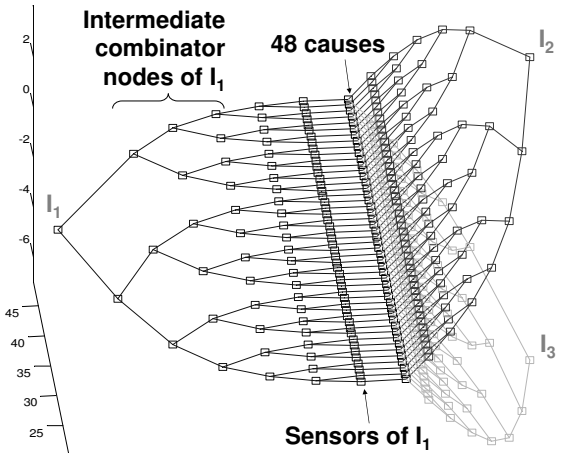

Figure 6: System of 48 causes and 3 effects, where causes and effects are fully connected

To propagate evidence (observations) through a bayesian network and to calculate the resulting posterior probabilities for all nodes not directly observed there are different approaches:

- Sampling methods (e.g. Gibbs-Sampling - see [8]): these methods are often usable for net-

fines the probability for each state given all possible state-combinations of all parent nodes.

[7] In this example the tables given for the effect-node and all intermediate combinator-nodes represent a probabilistic OR operator. Using other settings for these tables, it is also possible to create "AND combinator" nodes or even "XOR combinator" nodes.

[8] it is even possible to give that information with a probability, e.g. saying "I have checked component A and I am 80% sure that it is ok" (such evidence is called soft-evidence - see [7] for a more detailed explanation)

works where the junction-tree approach is not applicable, but have the disadvantage that the results are only approximate and that they are more time-consuming than propagation in junction-trees. Even worse, for some probability distributions and/or evidences they will fail completely.

- Junction-Tree methods (e.g. HUGIN propagation - see [7]): these methods lead to the exact posterior probabilities but might be too complex for some network structures.
- Loopy-Belief-Propagation: This method is based on the approach introduced by Perl (see [9]) for tree-like-structures. While for tree-like networks the results will converge to the correct values, this cannot be guaranteed for "loopy" network-structures. Nevertheless, surprisingly often the results are quite good (see [10] and [11]).

3.1 Sampling Methods

Generally speaking, sampling works as follows:

First create a full configuration:

1. Start at the parentless nodes and create (sample) a state out of the prior probability distribution given for every such node.
2. Now the states for all nodes having only parents with states already known (sampled) could be sampled in turn from the conditional probability table for $P(A|Parents(A))$.
3. Redo the last step until a full configuration is found, i.e. until for every node, a state has been sampled.

Likewise, create N full configurations. From all configurations, estimate the posterior distributions for every node by taking the fraction:

$$P(A = a_1) \approx \frac{\#(A = a1)}{N} \tag{1}$$

If evidence is given, then all configurations which are not in accordance with the evidence are rejected. Thus sometimes nearly all configurations will be rejected. Gibbs Sampling reduces this problem:

After a starting configuration (which is in accordance with the evidence) has been found it resamples all Nodes $N-1$ times given the current states of all other nodes.

Unfortunately, the type of networks used in this paper together with the fact that normally the effects (not the causes) are observed, does not permit the use of sampling methods.

- First of all, malfunctions are rare cases, which means that diagnostic scenarios cover a tiny part of the whole distribution only. Hence, big numbers of sampled configurations are needed to estimate the posterior distributions.
- Second, it is very likely that the initial configuration is not representative for the whole distribution, because the probability that any component is sampled to be faulty is very low, i.e. most probably the sampled configuration represents a false alarm of the set indicators.
- And finally, it is nearly impossible for the gibbs-sampler to escape from this "false alarm scenario" and get to a scenario where one or even more components are assumed to be faulty. The naive sampling method on the other hand will have to reject nearly all sampled configurations.

3.2 Junction-Trees

A Junction-tree is a calculation-model built from a bayesian network. In a junction tree cliques of nodes of the original bayesian network forming the nodes of the junction tree. These clique-nodes are connected by undirected links, such that a tree-like structure is created[9]:

Definition 4 (Tree)
In a tree, between every two nodes A and B, there is exactly one path connecting A and B, where, in such a path, each link could occur only once.[10] □

The links in a junction tree are called separators. A separator represents the subset of nodes included in both connected cliques.

Definition 5 (Separator)
Let $S_{1,2}$ be a separator connecting two Cliques C_1 and C_2, where C_1 contains nodes $\{C_1\}$ and C_2 contains nodes $\{C_2\}$ respectively, then $S_{1,2}$ contains the nodes $\{S_{1,2}\} = \{C_1\} \cap \{C_2\}$ □

To be valid, the junction-tree must meet the following condition:

Definition 6 (Junction-Tree)
Every node A which occurs in two Cliques C_1 and C_2 ($A \in \{C_1\} \wedge A \in \{C_2\}$) must also be contained in all cliques (and separators) on the path between C_1 and C_2[11]. □

[9] or a forest of disconnected trees, which can be seen as single junction trees each

[10] Notice: This is a definition usable for structures of undirected links (we do not need terms like "root" or "leaf")

[11] Remember: since it is a tree there is exactly one path

The latter condition could be achieved by moralizing and triangulating the bayesian network and take the cliques from the resulting structure.

Definition 7 (Moralized Graph)
To create a moralized graph from a bayesian network, all parents of common children get connected by an (undirected) link if they are not already directly connected. Afterwards, all originally directed links get substituted by undirected links. □

Definition 8 (Triangulated Graph)
A graph is called triangulated if for every cyclic path over more than three nodes there is a "shortcut" link which is not part of the path and which is directly connecting two nodes on the path. □

Definition 9 (Clique)
The cliques are formed by the largest sets of fully connected nodes of a moralized and triangulated graph. □

Definition 10 (Largest fully connected Subsets)
A subset of nodes is fully connected if for every two nodes A and B in that subset there exists a link connecting A and B. Such a subset is called the largest subset if no node could be added without violating the previous condition. □

One method for triangulating a (moral) graph and identifying the cliques is the "elimination-algorithm". This algorithm works as follows:

1. Eliminate one node A from the graph and connect all neighboring nodes of A. A and all its neighbors forming a new clique unless this clique would be a subset of another clique.
2. Go on with step 1 until all nodes have been eliminated.

The junction-tree then is formed by using the largest separators which do not violate the definion of a tree given before.

Since for every clique C_i the joint probability table $P(\{C_i\})$ is created, the aim is to keep the cliques as small as possible. Therefore the quality of a junction-tree depends on the triangulation which in turn depends on the order in which nodes were eliminated. Finding the best elimination-order is NP-hard for general network structures. A method to find the best triangulation for our restricted type of networks is subject to ongoing analysis.

Figure 7 illustrates the best elimination order found so far. Shown are the ranks for all nodes used to determine the elimination order. Lower ranks must be eliminated before higher ranks. The rank of the cause-nodes depends on the number of effect-nodes. Unfortunately, this algorithm only works for structures where all indicators are connected to all effects. To achieve this condition "dummy" sensor- and intermediate connector-nodes can be used. A dummy sensor node has 1 state instead of two and has no effect on the probability distribution modeled by the bayesian network. For a sparse connected structure, a normal one-step-look-ahead approach might find a better elimination order (see figure 8, small detail-plot) - for complex networks the AST-algorithm is much better (see figure 8, big plot).

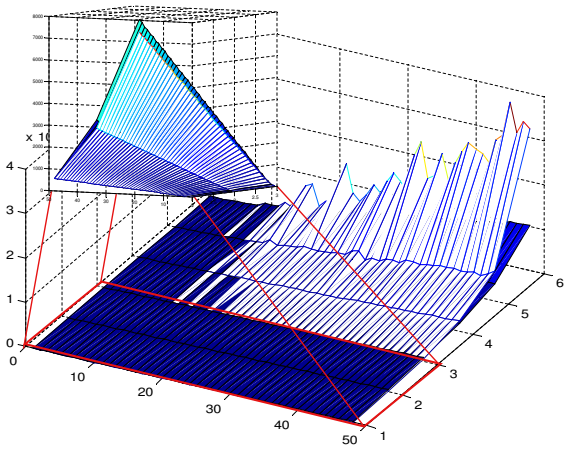

Figure 8: Number of parameters of a junction tree for the currently best triangulation (solid plot) vs. the number of parameters of a normal one-step-look-ahead elimination

Nevertheless, for very complex networks, namely where many causes are connected to many effects, this approach will fail, even if a perfect triangulation could be found. In figure 9, the resulting number of parameters for the best triangulation algorithm found so far is shown. It can be seen that the number of parameters grows linear with the number of causes, but exponential with the number of effects. This makes junction-trees inapplicable for situations where more than 8 effects are connected to a set of common causes.

3.3 Loopy Belief Propagation

Loopy belief propagation seems to be a promising solution for networks where junction-trees aren't applicable any more. Most important: the calculation-model has the same size as the underlying bayesian network[12].

For a loopy belief propagator we also create a structure of cliques and separators. In contrast to the junction-tree approach, for every node and its parents, one clique is created (again cliques that are

[12] in fact it is even slightly smaller

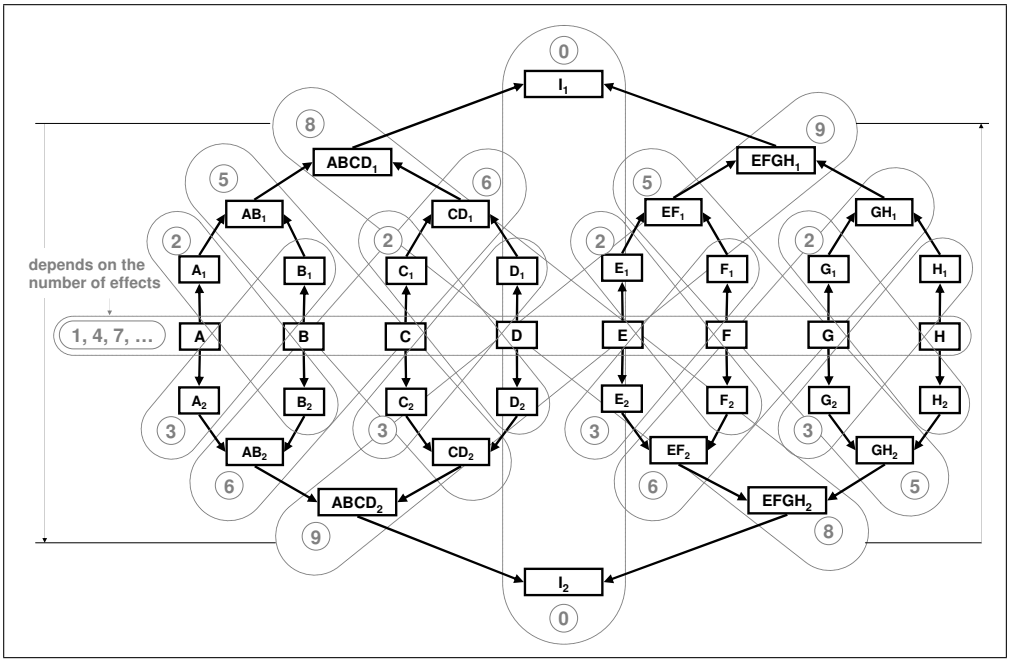

Figure 7: Elimination ranks for the adaptive stable triangulation algorithm (AST)

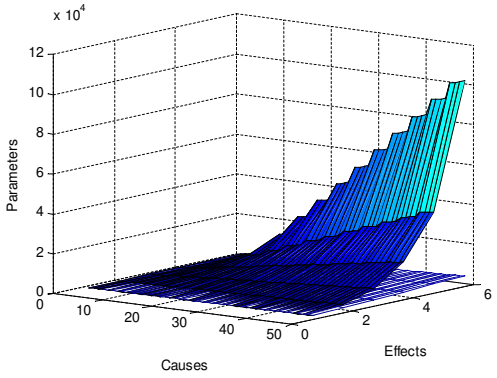

Figure 9: Number of parameters of a junction tree for the currently best triangulation (solid plot) vs. the number of parameters of the underlying bayesian network (mesh plot)

subsets of other cliques could be omitted). If two nodes A and B are connected, the cliques containing these nodes (e.g. C_1 and C_2) are also connected by a separator-link. The resulting structure is loopy if the underlying bayesian network is loopy.

Definition 11 (Loopy Graphs)
A graph is called "loopy" if it contains at least one path containing more than one node with the same start and end node. The direction of links is ignored here! □

A first analysis has shown that the convergence is fast and errors are tolerably small. This has to be evaluated in more detail, especially the circumstances leading to the maximum error are not yet fully understood.

3.4 Hybrid Solution

A hybrid type solution might be favored in the end: using junction-trees for clusters of the system with low complexity and loopy belief propagation for clusters of high complexity...

4 Result Post-Processing

Knowing the most likely cause of a malfunctioning system is one thing, but sometimes not enough to make the optimal decision. Imagine the following situation: A component A is the most probable defect ($P(A = faulty) = 80\%$) and there is another

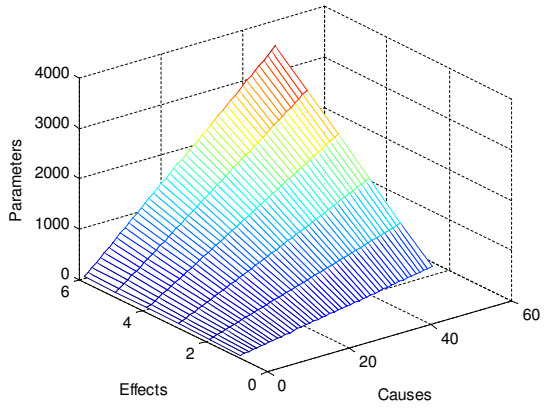

Figure 10: Number of parameters of a loopy belief propagator, where all causes are connected to all effects

component B also suspected but with a much lower probability ($P(B = faulty) = 10\%$). Normally it is best to check A first, but if that would be very expensive, say 100,-€, while checking B is cheap, say 10,-€, then things might change.

Formally the expected loss EL of checking a component X is defined by

$$EL(X) = P(X = ok) * Cost(X) \qquad (2)$$

If one wants to find the cause of a malfunction with a minimum loss, one should start with the component with the minimum expected loss. The cost could also be defined as a cost of time instead of money, if the target is to find the cause(s) as fast as possible.

5 Fault Isolation with Bayesian Networks

In this section some examples are presented. Starting with those examples which are straight forward - i.e. the results are essentially the same as for an approach using Boolean logic - except that each suspicion is quantified by a probability. Afterwards an example is presented where the Bayesian and the Boolean approach differs substantially, followed by an extended version where sensors are not restricted to be binary anymore. Finally it is shown that the ability to identify all possible configurations of multiple faulty components at once, which is one big advantage of the Boolean approach, could be realized also in the Bayesian approach, at least for the one most probable such configuration.

5.1 Straight forward Examples

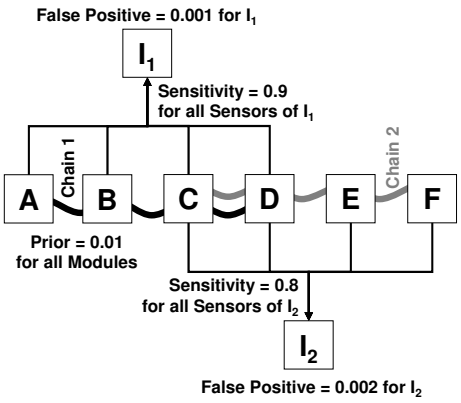

Figure 11: Setting of parameters for a system with two effects and six causes

Figure 11 shows an example with two indicators I_1 and I_2 with two common causes C and D. The prior probabilities of all causes are equally chosen to be 1%. Indicator I_1 is slightly better than I_2 because the false positive probability is less for I_1 (0.1% vs. 0.2%) and the sensitivities are higher for I_1 (90% for all causes vs. 80%).

	Indicators		Causes					
	I_1	I_2	A	B	C	D	E	F
1:	?	?	1	1	1	1	1	1
2:	?	?	(100)	1	1	1	1	1
3:	(0)	(0)	0.1	0.1	0.02	0.02	0.2	0.2
4:	(100)	(0)	40	40	8	8	0.2	0.2
5:	(100)	(0)	(0)	66	13	13	0.2	0.2
6:	(100)	(0)	(100)	1	0.2	0.2	0.2	0.2
7:	(100)	(100)	2	2	49	49	2	2
8:	(100)	(100)	48	48	(0)	(0)	45	45
9:	(100)	(100)	(100)	1	(0)	(0)	45	45
10:*)	(100)	(100)	40	4	32	32	41	4

*) Priors for A and E changed from 1% to 10%

Table 1: Example settings for the system shown in fig 11 (values in brackets are observed)

Descriptions referring to table 1

1. If nothing has been observed, the Bayesian network just gives the prior probabilities for all causes (i.e. 1%).

2. If A is known to be faulty, while nothing else is known, the prior probabilities for all other causes still are 1% - i.e. causes are independent of each other.

3. If both indicators are observed and none of them shows any problem, then the priors fall to 0.1% for A and B, to 0.2% for E and F[13] and down to 0.02% for C and D[14]

[13] Remember I_1 is a better indicator than I_2

[14] Notice that C and D having two indicators saying "everything is fine", while all other causes having just one indicator

4. If only I_1 reports a malfunction, then A and B are the most probable causes (40% each) while C and D get released by I_2 (i.e. I_2 says "C and D are o.k."). E and F aren't affected at all.

5. If, in addition, A has been observed to be ok, then B is the most probable cause (66%).

6. If A has been observed to be faulty instead, then B gets released because A fully explains why I_1 has detected a malfunctioning.

7. Now both indicators are set. In this situation, C and D are the most probable causes, because only C or D could each alone explain why both indicators are set. All other explanations would need two causes (e.g. A and E) to be faulty at the same time.

8. If both most probable causes have been observed to be ok, then it must be in fact a "double fault". Now all remaining causes get high fault-probabilities (48% for A and B, 45% for E and F)

9. Now, if A has been checked and has been observed to be in fact faulty, then the probability for B falls to 1%, while the probabilities for E and F stay the same. That is because at least one of E and F must be also faulty to explain why I_2 is set.

10. If the prior probabilities for A and E are much higher (10% instead of 1%) then a "double fault" is more likely than the single fault explanations C or D.

5.2 Overruling Indicators

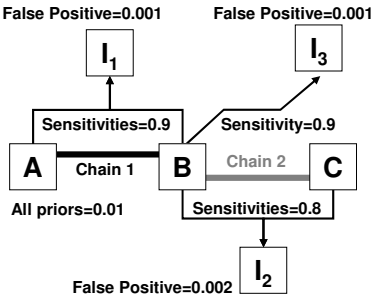

Figure 12: Setting of parameters for an example with three effects and causes

Imagine that in the system shown in Figure 12 the two indicators I_1 and I_2 are set while the third indicator I_3 is unset. Normally (using a Boolean approach) we would expect A and C to be identified as the most probable causes, because B is released by I_3. But even though I_3 is not worse than I_1 and even better than I_2 it is not yet "good" enough to favor a "double fault" of A and C over the assumption of a single faulty component B. That is why B is in fact the most likely cause - i.e. it is more likely that I_3 is wrong than a "double fault" of A and C.

5.3 Non-Binary Causes

Figure 13 shows an adaptation of the system already known from figure 11 . In this example the node C has three fault states instead of one (plus one state for being healthy). The first fault state here means a complete failure of the component C while the latter two fault states represent different partly failures, which are detected by two indicators with different sensitivities.

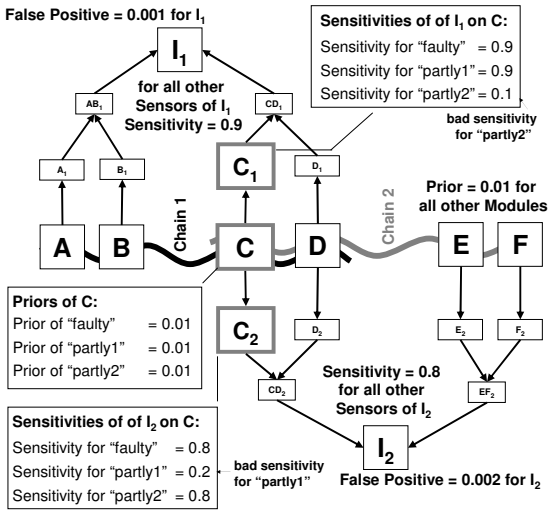

Figure 13: Setting of parameters for an example where one cause has three states

	Indicators		Causes							
	I_1	I_2	A	B	C			D	E	F
					f	p_1	p_2			
1:	(100)	(0)	30	30	6	25	1	8	0.2	0.2
2:	(100)	(100)	2	2	41	11	6	41	2	2

Table 2: Example settings for the system shown in figure 13 (values in brackets are observed)

The following descriptions refer to table 2

1. I_1 is set while I_2 is not. C can not be completely defect (because I_2 would have been set then), but it could be partly defect (because I_2 is not sensitive on the state $partly_1$).

2. If both indicators are set, C is likely to be completely faulty, because being only partly fault could not explain this scenario.

5.4 Finding Fault-Configurations

In the previous section we have seen examples where one single faulty component could fully explain the setting of indicators and examples where more than one component has been faulty. Unlike in the Boolean approach described in [2] we could not distinguish these situations - i.e. we don't know which components have to be faulty all together to fully explain the given scenario of set / unset indicators. Using junction-tree algorithms or loopy belief propagation it is possible to identify one such set in one shot[15] - and (fortunately) that turns out to be the most likely set of components. This method is called *Max-Propagation* (see [7]). In figure 14 an example is shown where the most likely configuration is that A and F are both faulty, after C and D have been inspected without success. Notice that, although A and B are the components which are most likely faulty (according to the normal propagation scheme), it is very unlikely that both of them are faulty. On the other hand it is also very unlikely that A is the only faulty component because a malfunction of A could not explain why indicator I_2 is set. Both could be seen - using the normal propagation scheme - by adding the observation that A is in fact faulty and propagate again. Doing so, one will see that the probability of B falls to a minuscule value, while the probability for E and F remains unaltered. Now, using *Max-Propagation* we are able to find that A and F are most likely both faulty in one single propagation.

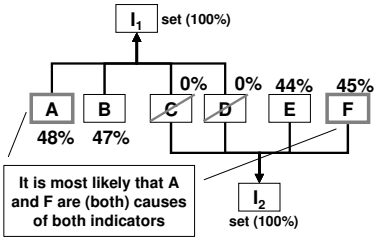

Figure 14: A double fault is the most likely explanation (after C and D were observed to be o.k.)

6 Learning from Data

As mentioned in section 2.3 the structure as well as the probabilities could be determined from technical sources like wiring diagrams, failure rates, and the like. In this section we will discuss another approach: learning structure and probabilities from data.

[15] just as expensive as a normal propagation-run

6.1 Learning the Probabilities

Since we use extra nodes which are pure "virtual" there will never be complete data for all nodes. Different approaches for learning parameters from incomplete data are available. One of it is the "Expectation Maximization Algorithm" (EM). It works as follows:

Init: Set all CPT's (which should be adapted) to randomized distributions, which are not containing the values 0 or 1 and which are not uniformly.

Expectation: Set all data available for one time-step t to the network and propagate the information through the network. Repeat that for all time-steps of the given dataset and take the sum of all resulting posterior distributions in contingency tables for each node.

Maximization: Set the conditional probabilities by normalizing the contingency tables for each node.

Iteration: Repeat from step 2 for a fixed number of iterations or until some convergence criteria is met.

This could be used for the type of networks described in this paper. The algorithm just won't adapt the probabilities which are fixed anyway - i.e. those that are set to 0 or 1.

6.2 Learning the Structure

Learning the structure is also possible. Several criteria for rating structures are known (e.g. [12]). A greedy search algorithm (also described in [12]) could be used. In our case we won't add single links but incorporate a connection from a cause to an effect over several intermediate nodes (sensors and intermediate connectors). Then we rate all possible new connections independently and take the best of them unless it doesn't improve the model. Since data is incomplete due to the "virtual" intermediate nodes, we have to learn the parameters by using the EM-algorithm described before in section 6.1 for each adaptation we make before we can rate it.

7 Special

This section shows some special cases, to which the system could be easily adapted to.

7.1 Logical Operations

The conditional probabilities shown in figure 5 for the indicator node and the intermediate combinator nodes representing an OR-combination of two parent nodes. The top indicator node includes the false positive probability, while the intermediate nodes are purely logical.

This represents an indicator which is set if any of the connected causes is true. This might not be adequate for all systems - e.g. some indicators might be set only if all of the connected causes are true. It is possible to set the probabilities such that the node performs an AND or even a XOR combination of its parent nodes. The conditional probabilities are shown in figure 3

AND

		I	
A	B	set	$unset$
set	set	1	0
set	$unset$	$FP^{*)}$	$1-FP$
$unset$	set	$FP^{*)}$	$1-FP$
$unset$	$unset$	$FP^{*)}$	$1-FP$

XOR

		I	
A	B	set	$unset$
set	set	$FP^{*)}$	$1-FP$
set	$unset$	1	0
$unset$	set	1	0
$unset$	$unset$	$FP^{*)}$	$1-FP$

*) FP is the false positive probability for top indicator nodes and 0 for intermediate combinator nodes.

Table 3: AND-combinator (top) / XOR-combinator (bottom)

7.2 Directly dependent Causes

As we have seen from the examples given in section 5.1 table 1 case 2 that causes are independent of each other. There might be systems where a malfunction of one or more components might directly lead to a malfunction of another component. This could be modeled the same way relations of indicators and causes are modeled - e.g. a component E which probably show a malfunction if component A or B or C or D is faulty could be modeled by replacing I_1 by E in figure 5. There is one important limitation inherent to that model: as an indicator such a cause could have two states only.

8 Dynamic Systems

Until now all systems were assumed to be "static" - i.e. all information used by a diagnostic system is created by making a "snap-shot" of the states of the system (states of the indicators and already checked modules). For highly dynamic systems we might want to use the information from previous time-steps as well. To account for dynamic systems we extend the approach developed so far by using two additional probabilities for each module. Now for each module three probabilities are required:[16]

- A failure probability, which is the probability for a healthy module to show some malfunctioning in the very next time step.
- A regeneration probability, which is the probability for a faulty module to be ok again in the very next time step.
- The prior probabilities already known, needed for the very first time step where modules haven't got any ancestors in previous time slices.

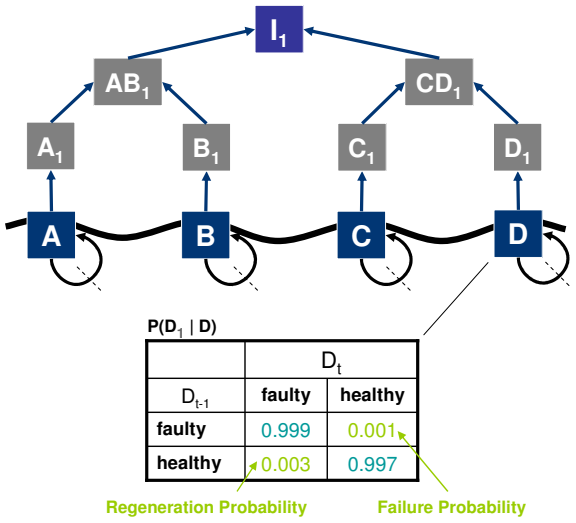

Figure 15: Additional parameters for dynamic networks

8.1 Dynamic Propagation

Different approaches have been discussed for creating junction trees for dynamic bayesian networks. One solution is to "unroll" the bayesian network for

[16] Both new probabilities (the failure and the regeneration probability) highly depend on the sampling rate and will be tiny for most systems.

a given number of time steps and create a junction tree for that network. Unfortunately this approach has some drawbacks:

- The number of time steps must be known before any propagation could be done
- The propagation will calculate over all time-steps even if the observations and the nodes of interest all lying in a defined time-window (which moves from time-step to time-step)

Another solution is to create a dynamically adjustable junction tree, which contains the nodes of a given time-window only. To create such a "dynamic junction tree", all the nodes having dynamic links must be combined into one single clique (the "dynamic clique") - otherwise the resulting structure wouldn't be a tree. Unfortunately this is impossible for systems having many time dependent nodes, because the dynamic clique grows exponentially with the number of contained nodes. Therefore a loopy belief propagation scheme should be used, even if the module for a single time slice is a real junction tree.

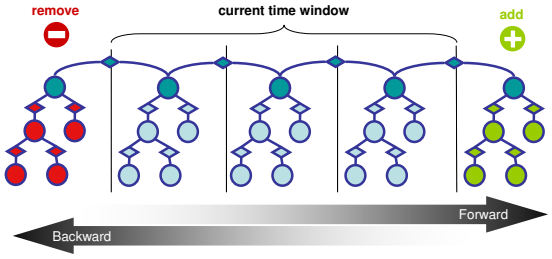

Figure 16: A dynamic junction tree

9 Conclusions

Using probabilities to represent the uncertainties, naturally occurring in diagnostic applications, is a promising way to overcome some of the limitations of traditional diagnostic systems. We have presented a method to create a compact consistent probability model representing a technical system as a bayesian network. A challenging task for the future will be to find a propagation algorithm which provides the best tradeoff between accuracy and complexity for such Bayesian networks.

References

[1] Peter Tondl, Patrick Rammelt, and Ulrich Siebel. State based Diagnostics of System Internal Fault Origins by using Boolean Rules. In *Diagnose in mechatronischen Fahrzeugsystemen IV*. Bäcker, Unger, 2011. (978-3-8169-3068-6).

[2] Peter Tondl, Patrick Rammelt, Oliver Berger, and Ulrich Siebel. An Efficient Diagnostic Algorithm based on Boolean Rules. In *Diagnose in mechatronischen Fahrzeugsystemen V*. Bäcker, Unger, 2012. (978-3-8169-3149-2).

[3] Klaus Lange and Ulrich Siebel. Ein neuer Diagnoseansatz für moderne Fahrzeugsysteme. *ZfAW – Zeitschrift für die gesamte Wertschöpfungskette Automobilwirtschaft*, 3:72–76, 2009.

[4] F V Jensen, C Skaanning, and U Kjaerulff. The SACSO system for troubleshooting of printing systems. In *in Seventh Scandinavian Conference on Artificial Intelligence, Frontiers in Artificial Intelligence and Applications*, pages 67–79. IOS Press, 2001.

[5] Olaf Krieger, Andreas Breuer, and Tobias Müller. Intelligente Fahrzeugdiagnose mit variabler Prüfstrategie, 8 2008.

[6] Olaf Krieger, Andreas Breuer, Klaus Lange, Tobias Müller, and Thomas Form. Wahrscheinlichkeitsbasierte Fahrzeugdiagnose auf Basis individuell generierter Prüfabläufe, 8 2008.

[7] F V Jensen. An Introduction to Bayesian Networks, 1996.

[8] Addendum To Manual, David Spiegelhalter, Andrew Thomas, Nicky Best, and Wally Gilks. Bayesian inference Using Gibbs Sampling.

[9] J Pearl. Probabilistic reasoning in intelligent systems: Networks of plausible inference, 1988.

[10] K P Murphy, Y Weiss, and M I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *In Proceedings of Uncertainty in AI*, pages 467–475, 1999.

[11] Janneke H. Bolt and Linda C. van der Gaag. On the Convergence Error in Loopy Propagation.

[12] D Heckerman. A tutorial on learning Bayesian networks. Technical report, Communications of the ACM, 1995.