Model-based Probabilistic Diagnosis in Large Cyberphysical Systems

Giso Dal[1], Arjen Hommersom[2], Guus Grievink[3], and Peter J.F. Lucas[4]

[1,3,4] *EEMCS Faculty, University of Twente, Drienerlolaan 5, 7522 NB Enschede, The Netherlands*
gisodal@gmail.com, g.grievink@student.utwente.nl, peter.lucas@utwente.nl

[2] *Open University, Valkenburgerweg 177, 6419 AT Heerlen, The Netherlands*
arjen.hommersom@ou.nl

ABSTRACT

Model-based diagnosis is concerned with diagnosing faults or malfunction of real-world physical or cyberphysical systems using a model of the structure and behavior of the systems. As cyberphysical systems can be extremely large and complex, and the associated computational models will be then equally large and complex, they impose a hard to beat challenge on the computational feasibility of reasoning with such models. When such a model is able to handle the uncertainty associated with diagnostics, giving rise to probabilistic model-based diagnostics, the computational feasibility becomes even harder. This paper: (1) proposes a novel graphical method underlying model-based diagnostics; (2) demonstrates experimentally how a novel, by the authors developed architecture of partitioned positive weighted model counting, is able to handle exact inference to answer a variety of probabilistic queries regarding the health status of a cyberphysical system. Results obtained are well within acceptable time bounds.

1. INTRODUCTION

Cyberphysical systems combine and integrate physical and computational processes often with a special role for sensor information (Lee, 2008). Nowadays, because of the explosive rise in the role of embedded software in physical systems, there are many of such systems, for example industrial printing or vending machines. In particular in a commercial setting, such machines are desired to experience the least possible downtime, as being out of order usually has undesirable, often financial, consequences for the users. Thus there is a need to find the causes of malfunction as quickly as possible, a process usually referred to a *diagnosis*, a terms taken from the field of medicine (Lucas, 1996); the terms *troubleshooting* and *fault-finding* are also often used in engineering. In the case of cyberphysical systems faults or defects concern physical or software components, or possibly their interaction. The purpose of the diagnostic process consists of automated fault finding followed by repair or assisting technicians in a repair job on site (Grievink, 2022).

Automated computer-based diagnosis in engineering has a long-standing tradition, where in particular fault tree analysis is a commonly used technique (Ruijters & Stoelinga, 2015). However, other automated fault detection and analysis methods have also been developed (Dowdeswell, Sinha, & MacDonell, 2020). In the present paper we depart from a framework developed in the 1980s by Johan de Kleer, and which is known as *model-based diagnosis*, MBD for short (de Kleer & Williams, 1987). The adjective 'model-based' comes from the principle that with the design of a machine one possesses already valuable knowledge about its structure or architecture and its functional components and their interactions before the machine is actually produced, purchased, and employed in practice. This knowledge can be put to use in a diagnostic setting.

De Kleer's method of model-based diagnosis is based on comparing qualitative predictions of behavior of a *model* of a given machine with actual observations, which explains why it is also called *consistency-based diagnosis* (CBD) (Reiter, 1987). CBD is traditionally seen as a kind of symbolic or logical assumption-based reasoning (Genesereth & Nilsson, 1987). However, even in the early days of MBD it was realized that probabilistic information could play a role in improving the accuracy of diagnostic solutions (de Kleer, 1991). It was subsequently proved by Judea Pearl that MBD can be mapped to the multivariate probabilistic representation of Bayesian networks (Pearl, 1988). The advantage of Bayesian networks as a formalism of model-based diagnosis is that in principle uncertain, probabilistic knowledge about the occurrence of faults can be integrated and also learned from data.

However, a well-known problem of MBD, and probabilistic MBD is no exception, is that models can be very large and thus inference is often infeasible. There can be little doubt that with the large and complex cyberphysical systems developed today, and even more so in the future, building and performing inference with such models will hit computational obstacles. During the last few years we have been working on moving the boundaries of probabilistic inference by developing a novel framework referred to as *partitioned positive weighted model counting* (that can also be parallelized), inspired by the success of model counting in software verification (Dal & Lucas, 2017; Dal, Laarman, Hommersom, & Lucas, 2021). This framework was shown in various papers to be superior to other probabilistic methods (Dal et al., 2021; Dal, Laarman, & Lucas, 2023). This made us wonder whether it might be a good candidate for model-based diagnosis of large cyberphysical systems. Positive weighted model counting exploits symmetries in probability tables, which typically also occur in models used in MBD.

The main contributions of this paper are as follows:

- A new representation of compositional Bayesian networks that supports developing large probabilistic model-based systems from specifications of system components;
- A novel method of probabilistic model-based diagnosis, we call it *Bayesian* model-based diagnosis, that properly takes into account the dependences between components in MBD, different from an earlier developed and limited method (Grievink, 2022);
- Experimental evidence that our publicly available software tool PARAGNOSIS [1] (Dal et al., 2023), implementing partitioned weighted model counting, supports solving diagnostic problems of systems with different size and complexity, including very large and complex ones.

The organization of this paper is as follows. First, in Section 2, some basic principles of consistency-based diagnosis are introduced, follows by a compact summary of the method of weighted model counting, and the mapping of MBD to Bayesian networks. As our work on partitioned positive weighted model counting has been extensive published, we refer for details about how it works to those publications (Dal & Lucas, 2017; Dal, Michels, & Lucas, 2017; Dal et al., 2021, 2023). In Section 3 the compositional method of assumption-based Bayesian model-based diagnosis is developed. Experimental results are summarized in Section 4, which is followed by conclusions and a discussion of the results in Section 5.

2. BACKGROUND

2.1. Consistency-based Diagnosis

Given specific input and output of a cyberphysical system, the output of the model of the system is compared with the observed output of the (real) system. A discrepancy between these two indicates a fault or malfunction in the actual system and explains the name 'consistency-based diagnosis' (Reiter, 1987). Below, some of the common definitions that occur in the literature on CBD are repeated and adapted, where in particular (de Kleer, Mackworth, & Reiter, 1992) is followed.

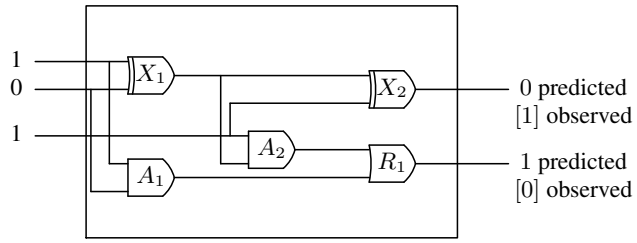


Figure 1. Full adder with inputs and observed and predicted outputs. Here, Obs $= \{\text{in1}(X1) = 1, \text{in2}(X1) = 0, \text{in1}(A2) = 1, \text{out}(X2) = 1, \text{out}(R1) = 0\}$.

A *diagnostic problem* DP is defined as a system SYS together with a *set of observation* Obs: DP $= (\text{SYS}, \text{Obs})$. A *system* SYS consists of a *system description* SD and a set of *components* Comps: SYS $= (\text{SD}, \text{Comps})$. The system description defines the normal behavior of components and how these components are connected by means of first-order logical sentences. Given SYS, a *diagnosis* $D \subseteq$ Comps is defined as a *subset-minimal* set of components that, when behaving abnormally, explains the observation of a faulty system. Formally, a diagnosis is defined as a subset-minimal set $D \subseteq$ Comps such that:

$$\text{SD} \cup \text{Obs} \cup \{\text{Ab}(c) \mid c \in D\} \cup \{\neg\text{Ab}(c) \mid c \in \text{Comps} \backslash D\} \not\models \bot \quad (1)$$

where 'Ab' is the abnormality predicate that indicates that a component c behaves abnormally (and thus $\neg$Ab indicates normal behavior) and $\bot$ is falsum (the left-hand side of $\not\models$ is consistent).

Example 2.1 (Adapted from (Reiter, 1987))**.** *Consider the logical circuit depicted in Fig. 1, which represents a full adder, i.e. a circuit that can be used for the addition of two bits with carry-in and carry-out bits. This circuit consists of two* AND *gates (*$A1$ *and* $A2$*), one OR gate (*$R1$*) and two exclusive-*OR *(*XOR*) gates (*$X1$ *and* $X2$*);* $\text{Comps} = \{A1, A2, X1, X2, R1\}$*. The input and output of components* c *are denoted as* $\text{in}(c)$ *and* $\text{out}(c)$*, respectively.*

The behavior description SD *consists of the following axioms:*

$$\neg\text{Ab}(c) \rightarrow \text{out}(c) = \text{and}(\text{in1}(c), \text{in2}(c)), \textit{ for } c \in \{A1, A2\},$$
$$\neg\text{Ab}(c) \rightarrow \text{out}(c) = \text{xor}(\text{in1}(c), \text{in2}(c)), \textit{ for } c \in \{X1, X2\},$$
$$\neg\text{Ab}(c) \rightarrow \text{out}(c) = \text{or}(\text{in1}(c), \text{in2}(c)), \textit{ for } c = R1.$$

These logical rules describe the normal behavior of each individual component (gate).

[1] https://github.com/gisodal/paragnosis

The component connections are described as follows:

$$\begin{aligned} \text{out}(X1) &= \text{in2}(A2) & \text{out}(X1) &= \text{in1}(X2) \\ \text{out}(A2) &= \text{in1}(R1) & \text{in1}(A2) &= \text{in2}(X2) \\ \text{in1}(X1) &= \text{in1}(A1) & \text{in2}(X1) &= \text{in2}(A1)\ . \\ \text{out}(A1) &= \text{in2}(R1) \end{aligned}$$

With the observations Obs *as indicated in Fig. 1 it is clear that when assuming the empty diagnosis, $D = \emptyset$ — all components are behaving normally — 1 will give an inconsistency, as also indicated in the figure (predicted and observed outputs differ). There are multiple solutions for the diagnostic problem in this case. For example, $D = \{X1\}$, $D' = \{X2, R1\}$, and $D'' = \{X2, A2\}$ are diagnoses.*

2.2. Weighted Model Counting

2.2.1. Bayesian Networks

A *Bayesian network* (BN) $\mathcal{B} = (G, P)$ is a directed acyclic graph $G = (V, A)$ that associates 1–1 random variables X_v to nodes $v \in V$ in the graph (Pearl, 1988). The directed edges $(v, w) \in A$ represent conditional (in)dependence assumptions and P stands for a joint probability distribution of the set of variables X_V defined as follows:

$$P(X_V = x_V) = \prod_{v \in V} P(X_v = x_v \mid X_{\pi(v)} = x_{\pi(v)}) \quad (2)$$

Thus, a BN is defined in terms of a (family of) conditional probability distributions of $X_v \in X_V$ given the variables corresponding to the parents $\pi(v)$ of $v \in V$ in the graph, i.e., $X_{\pi(v)}$, specified as $P(X_v \mid X_{\pi(v)})$, called *conditional probability tables* or CPTs for short in the following.

Posterior probability distributions of the form

$$P(X_U \mid \text{Evidence}), \quad (3)$$

with 'Evidence' a set of *observations* or measurements concerning particular variables $X_v \in X_V$, with typically $v \notin U$, $U \subseteq V$, can be computed based on the specification of a BN —a process called *probabilistic inference* or *reasoning*— using common axioms of probability theory. However, for real-life networks advanced algorithms are required as the computation is NP-hard in general and often quite intensive for real-life networks (Koller & Friedman, 2009).

By exploiting the conditional independence assumptions, BNs represent concise factorizations of a joint probability distribution. The size of the factorization has direct implications toward the cost of probabilistic inference. A more expressive model must be used to in order to exploit properties of CPTs (Chavira & Darwiche, 2008). A prominent way of achieving this is to find a more concise and canonical representation such as a Binary Decision Diagram (BDD) (Bryant, 1986). Compiling a BN to a decision diagram (DD) representation is commonly referred to as *knowledge compilation* (Darwiche & Marquis, 2002), or simply compilation.

Table 1. Three examples of models of the encoding of variable X and associated probability distribution.

	Models					Associated probability
1	x_1	$\overline{x_2}$	$\overline{x_3}$	ω_1	$\overline{\omega_2}$	$W(\omega_1)W(\overline{\omega_2})$=$0.8 \cdot 1$ =0.8
2	$\overline{x_1}$	x_2	$\overline{x_3}$	$\overline{\omega_1}$	ω_2	$W(\overline{\omega_1})W(\omega_2)$=$1 \cdot 0.1$ =0.1
3	$\overline{x_1}$	$\overline{x_2}$	x_3	$\overline{\omega_1}$	ω_2	$W(\overline{\omega_1})W(\omega_2)$=$1 \cdot 0.1$ =0.1

2.2.2. Encoding

Prior to compiling a BN to a DD, we require an encoding to transition from the multi-valued domain of discrete random variables to the Boolean domain. There are multiple ways to do this. We choose to first translate a BN to a Boolean formula with dedicated variables to represent probabilities (Chavira & Darwiche, 2008; Dal & Lucas, 2017).

Conjunctive Normal Form (CNF) from logic, where formulas consist of conjunctions of subformulas of literals with only disjunctions, called *clauses*, is commonly used to facilitate compilation. We create for every $X_v \in X_V$ a set of atoms $a(X_v) = \{x_1, \ldots, x_n\}$. Semantically, $x_i \in a(X_v)$ represents X_v being equal its i^{th} value. In addition, an atom ω_j is introduced for every unique probability in X_v's CPT, i.e., ω_j can refer to multiple distinct entries in X_v's CPT if they represent the same probability. A clause is introduced for each entry of the CPT, with an ω_j atom that has a weight $W(\omega_j)$ that is linked to the actual probability, and $W(\overline{\omega_j}) = 1$. Finally clauses are added to prevent inconsistent representations, such as making sure that a variable cannot get multiple values at the same time. This is illustrated by an example (Dal & Lucas, 2017)

Example 2.2 (Bayesian Network encoding)**.** *Let BN $\mathcal{B}$ be defined for variables $\{X, Y\}$ with factorization $P(X, Y) = P(Y \mid X)P(X)$. For simplicity's sake, we focus on just variable X; X has three values, thus the CPT has 3 entries, in this case only two distinct. To encode X and its probabilities we create atoms $a(X) = \{x_1, x_2, x_3\}$. The atom ω_1 is introduced for $X = 1$, ω_2 for $X = 2$ and $X = 3$ (as they have the same probability), with $W(\omega_1) = 0.8$, $W(\omega_2) = 0.1$.*

The CNF representation is as follows:

$$(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \omega_1) \wedge (\overline{x_2} \vee \omega_2) \wedge (\overline{x_3} \vee \omega_2)$$

The first clause enumerates the possible values of X, whereas the second to fourth clause ensure that X (as a random variable) cannot have more than one value. The last three clauses link a probability to an actual value of X. The encoding includes the truth assignments (models) for variable X as shown in Table 1. Note that the weighted model count sums to 1.0 *for this selection of models. However, there are other*

models of this CNF, e.g., model $\{x_1,\overline{x_2},\overline{x_3},\omega_1,\omega_2\}$*, model* $\{\overline{x_1}, x_2, \overline{x_3}, \omega_1, \omega_2\}$*, etc. Only minimal models sum to* 1.0*, i.e., models with the largest number of negations.*

2.2.3. Compilation

Now that we have an encoding, we can consider its compilation to a *Weighted Positive Binary Decision Diagrams* (WPBDD) (Dal & Lucas, 2017). A WPBDD is an ordered BDD that represents a concise factorization of a Boolean formula f as a (rooted) directed acyclic graph with decision nodes, and two terminal nodes labeled with $\top$ (true) and $\bot$ (false). Each non-terminal node v is labeled with a Boolean variable $\text{var}(v) = x_v$ and has two children, $\text{high}(v)$ and $\text{low}(v)$, with a set of weight variables $\text{weights}(v)$ at the edge to node $\text{high}(v)$ (explaining the adjective 'positive' in WPBDD). Each root-terminal path contains a variable at most once, and in a particular total or partial order.

A CNF encoding as described above acts as an entry point for the language compiler (Dudek, Phan, & Vardi, 2020). Such compilers target different variations of DDs.

The respective DD is built using the typical bottom-up strategy (Bryant, 1986), by applying DD operations to construct a DD representing the encoded formula from the previous step. The process of compiling into a respective DD is by far the most expensive operation, compared to the inference step, which is linear in the size of the DD as desired.

2.2.4. Inference

Inference is performed through *Weighted Model Counting* on the DD, WMC for short (Chavira & Darwiche, 2008; Darwiche & Marquis, 2021). This process sums the weight of every truth assignment. In the decision diagram, these assignments are represented by paths and the weights by edge labels. Edges to nodes $\text{high}(v)$ and $\text{low}(v)$ are solid $\rightarrow$ and dashed $\dashrightarrow$, respectively (see below). Since these paths often overlap in the DD structure, inference through model counting is linear in the size of the target representation (Darwiche & Marquis, 2002).

Let's look at a WPBDD compilation and inference example. A WPBDD exactly represents the encoding provided. In order to perform inference we can trivially transform the logical circuit that the WPBDD represents into an arithmetic circuit.

Example 2.3 (Compilation and inference)**.** *Consider again variable* X *from Example 2.2. For the compiled DD the ordering for variable* X *is ordering* $x_3 \prec x_2 \prec x_1$*. Reduction rules specific to WPBDDs allow the removal of the* x_2 *node to further reduce its size. Each path from the root to the* $\top$*-terminal semantically implies evidence. There are three possible paths shown below. If we have evidence prior to traversing the compiled representation, we only consider the paths that are consistent with the evidence.*

Path	*Logic*	*Semantics*
$x_3 \rightarrow \top$	$\overline{x_1} \wedge \overline{x_2} \wedge x_3$	$X = 3$
$x_3 \dashrightarrow x_2 \rightarrow \top$	$\overline{x_1} \wedge x_2 \wedge \overline{x_3}$	$X = 2$
$x_3 \dashrightarrow x_2 \dashrightarrow x_1 \rightarrow \top$	$x_1 \wedge \overline{x_2} \wedge \overline{x_3}$	$X = 1$

To perform inference, we need to link to the probabilities that allows us to compute $P(X = 3) = 0.1$*, by the assignment* $(x_1, x_2, x_3) = (\bot, \bot, \top)$.

The tool ParaGnosis offers important ways to optimize compilation and inference by partitioning and parallelization (for details see (Dal et al., 2021, 2023)).

2.3. Mapping to a Bayesian Network

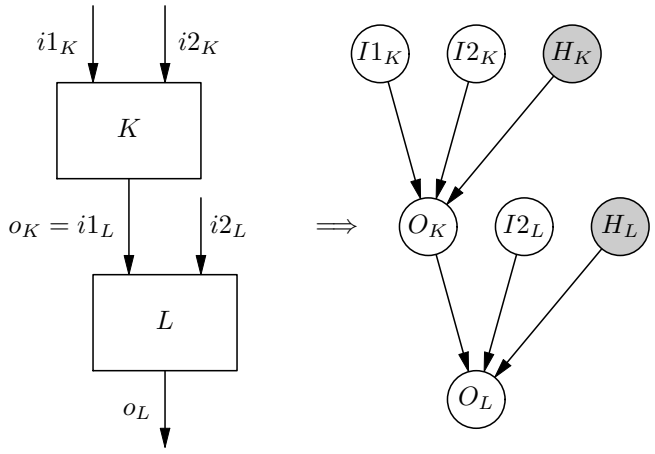
To add a probabilistic aspect to consistency-based diagnosis, a logical diagnostic problem can be mapped to a Bayesian network. There are different ways for translating a diagnostic problem into a *Bayesian diagnostic problem*. Two of these will be highlighted. One of these is the traditional method proposed by Pearl (Pearl, 1988), and implemented later by Srinivas (Srinivas, 1994), and the other is a more recent adaptation introduced by us. The latter one is used for this research, but since it is based on the traditional method both will be expanded upon.

2.3.1. Pearl's Method

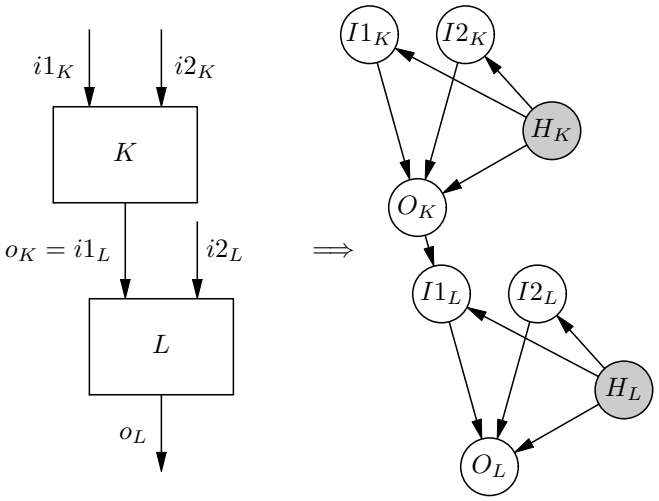
Following Pearl's method (Pearl, 1988), in Fig. 2a an abstract 2-component system has been translated into a Bayesian network. Each input and output of a component are modeled as nodes. A component is modeled as an output node that is the child of all its input nodes. Note that one of the inputs of component L is the output of component K and thus the output node of K is directly linked to the output node of L. Next to these, per component, a health node H (also called 'abnormality node' in the literature) is added as the parent of the output node. This node corresponds to the abnormality predicate in MBD and similarly indicates whether or not the component behaves abnormally: the abnormality literal of the traditional approach could be mimicked by assigning the values 'normal', corresponding logically to the assumption '$\neg\text{Ab}(c)$', whereas the value 'abnormal' would correspond to '$\text{Ab}(c)$'. However, a disadvantage of this approach is that health nodes and input nodes are independent; they only become conditionally dependent when a common child of input nodes or a descendant of the child is instantiated to a value (Pearl, 1988).

2.3.2. Method Based on Connected Health Nodes

The method which is used in the present paper no longer assumes that inputs and health nodes are independent. The health nodes support enforcing extra dependencies between the inputs and outputs, and when none of the children or descendants of the children of the input nodes are instantiated. This method is illustrated in Fig. 2b. Note that in this repre-



(a) Pearl's method with health nodes; health nodes shaded.



(b) New method with health nodes connecting inputs and output; health nodes are shaded.

Figure 2. Two methods of mapping of a simple system model with two components K and L to a Bayesian network.

sentation, all inputs are explicitly represented in the Bayesian network and the relationship between the output of the previous subpart (e.g. component K) and the next subpart (e.g. component L) is represented by an arc between the output node and one input node (e.g. the arc $O_K \rightarrow I1_L$). The conditional probability distribution of any connected output node O, $P(O \mid I_1, \ldots, I_n, H)$, is such that $P(O \mid I_1, \ldots, I_n, H = \text{normal}) \in \{0, 1\}$, dependent on the value of input variables I_k, with $k = 1, \ldots, n$.

2.3.3. Establishing a Bayesian Diagnosis

With the logical diagnostic problem mapped to a Bayesian diagnostic problem, probabilistic inference methods can be used to derive whether or not the components behave correctly as expected to form a diagnosis (Pearl, 1988).

Before such derivation can take place, first the evidence, consisting of observed values of inputs and specific outputs, should be included, which is analogous to the observations 'Obs' in MBD. With probabilistic inference methods, the posterior probabilities of each of the chosen health nodes can be calculated. Then for a given set of health variables H, a *diagnosis* is defined as follows:

$$D = \arg\max_{h} P(H = h \mid \text{Evidence}, \text{Health-assumptions}) \tag{4}$$

i.e., the assignment h to H with the maximum probability, where it is possible to condition on the health variables not included in H by 'Health-assumptions', the other health variables that are given an (assumed or observed) value. We call this process *assumption-based Bayesian model-based diagnosis*, or Bayesian MBD for short.

The same method can be used for Pearl's Bayesian-network structure of a diagnostic model. However, in that case it is mandatory to instantiate the last output variables to enforce dependence between input and health values and one can no longer simulate various flow schemes.

3. METHODOLOGY

To investigate whether weighted model counting with partitioning offers a suitable and fast algorithm for Bayesian model-based diagnosis, some Bayesian MBD models were designed. Unfortunately, it is virtually impossible to get access to large industrial MBD systems with their associated data for a publication, because of the industry's fear of disclosure of competition-sensitive information to the public domain. For this reason, we had to resort to designing an artificial model, that nevertheless was inspired by existing pipe systems as used in the chemical and oil industry.

The research question that is explored in the remainder of this paper is whether partitioned positive weighted model counting can effectively deal with large Bayesian MBD models in such a way that acceptable diagnostic results are obtained. For this purpose the PARAGNOSIS toolkit was implemented (Dal et al., 2023).

3.1. Basic Elements

Bayesian MBD requires the development of Bayesian-network models of systems, where the models consist of *components*, where some of those components are identical in nature, and compositional ways to merge these models together to build an overall model of a system. The result will be an abstraction of the real-world system that reflects both structure and behavior of the real-world system and that can be used for simulation purposes. In addition for MBD it is necessary to include *behavior modes*, e.g., whether the component is behaving normally or abnormally (other modes are sometimes also used) for each of the components that could be defective, which will be represented as *health variables*. Finally,

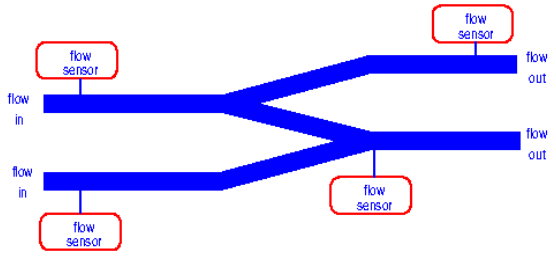


Figure 3. System of connected fluid pipes with one join, one split, and four flow sensors.

information about the behavior of a component is obtained through *sensors*.

3.2. Generating a System

Consider a pipe system, such as the one shown in Fig. 3, that consists of connected pipes, joints of pipes, and splittings of pipes (splits) that transport a fluid, e.g., water or oil, and have an input and output of fluid. The flow of the fluid is measured by means of *flow sensors* and is discretized into three states: 'maxflow', 'lowflow', and 'noflow'. Each pipe has one flow coming into it and one flow going out of it; if pipe x behaves normally:

$$\forall x(\text{PipeIn}(x) = v \rightarrow \text{PipeOut}(x) = v)$$

with v a free variable standing for a state. However, the actual health status of the pipe is ignored here, and assumed to be normal. Alternatively, if we want to take into account the health status and pipe x has a leak, the outgoing flow will be lower than the incoming flow:

$$\forall x((\text{PipeIn}(x) = \text{maxflow} \quad \wedge \quad \text{Health}(x) = \text{leak}) \rightarrow \text{PipeOut}(x) = \text{lowflow}))$$

whereas for normal health we get:

$$\forall x((\text{PipeIn}(x) = v \quad \wedge \quad \text{Health}(x) = \text{normal}) \rightarrow \text{PipeOut}(x) = v))$$

Thus, it is needed to include the health status as an additional condition. Similar logical specifications can be developed for the other elements of pipe systems, i.e., the joints and splittings.

As we use individual pipe components to develop (generate) pipe systems of various complexity and size, as is also done when developing real-life pipe systems, we will number pipe components from input to output flow of the entire system, in terms of two parameters: width (from left to right) and height (top to bottom) of the directed graph (starting with 1). We come back to this issue in Section 4.

A clear disadvantage is that uncertainty in the health status and the likelihood that a leak may give rise to low or no flow is missing in the logical representation. Bayesian model-based diagnosis will support representing this important aspect of diagnosis, i.e., the ability to deal with uncertainty, as will be discussed below.

3.3. Uncertainty and Bayesian-network Components

To model a system that incorporates uncertainty, we need design Bayesian-network components that correspond to the various parts of the real system. Based on the mentioned specifications in the section above, we distinguish pipe, join, and split Bayesian-network components. The health status of a pipe component is controlled by a health variable, called variously 'JoinHealth', 'PipeHealth', 'SplitHealth'. As above, the continuous flow is mapped to discrete values, we distinguish three flow values: maximal (maxflow), low (lowflow), or absent (noflow).

The three basic Bayesian-network components are depicted in Fig. 4, 5, and 6. These can be put together in various topologies giving rise to a plethora of pipe systems. In addition sensors can be added to components to measure the status of the flow in the individual pipes. Sensor readings will be used below as evidence in the experiments to diagnose faults in the different pipe systems. Software and figures have been generated using the R language (R Core Team, 2024), the R-library bnlearn (Scutari, 2024), with GeNIe Modeler[2] for producing graphical figures.

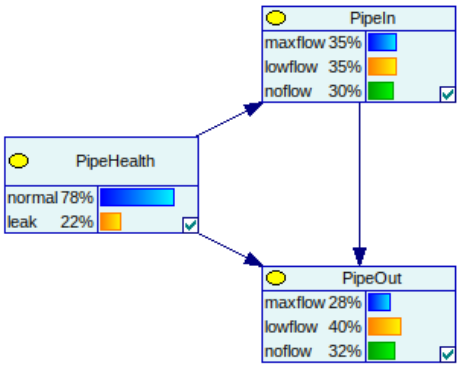


Figure 4. Pipe-shaped component; there is no need to model the actual pipe as it is sufficient to represent the input and output of the pipe and how these relate to each other by means of health modes.

3.4. Compositionality by Probability Distributions

Each pipe is modeled as a flow out of 'PipeOut' with one flow coming into 'PipeIn'. The prior probability of a pipe's health being normal is set to 0.8 ($P(\text{PipeHealth} = \text{normal}) = 0.8$). The distribution of the input of the pipe depends on the output of the previous component in a deterministic manner,

[2] https://www.bayesfusion.com/genie

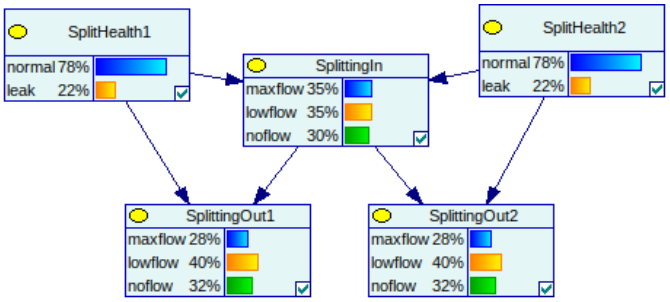


Figure 5. Split-shaped component consisting of a single input and in this case two outputs. In addition, each output is controlled by a health mode.

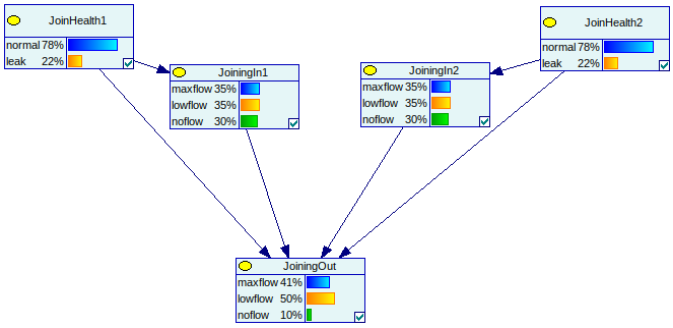


Figure 6. Join-shaped component consisting of two (or more) inputs and a single output that merges the inputs. In addition, the actual merge is controlled by one health node per input.

i.e., we do not consider defects between components. Since this structure is the same for every pipe, the CPT of each pipe is also identical such that the probability distribution

$$P(\text{PipeOut}_i \mid \text{PipeIn}_i = v, \text{PipeHealth}_i = w)$$

is the same for each i, but varies for specific values of v and w as shown in Table 2. In case the component behaves normally, there is no impact on the flow of the pipes, which reflects the logical specifications of Section 3.2. Furthermore, the CPT reflects the impact of a leaking pipe. Given a normal flow, if the pipe is leaking, there is a high probability the flow is reduced, and in severe cases, it may even lead to no flow in the output. If there is low flow in the input, we assume

Table 2. CPT for the output node of a pipe.

		PipeOut$_i$		
		maxflow	**lowflow**	**noflow**
PipeIn$_i$	**PipeHealth$_i$**			
maxflow	**normal**	1	0	0
	leak	0.1	0.8	0.1
lowflow	**normal**	0	1	0
	leak	0	0.8	0.2
noflow	**normal**	0	0	1
	leak	0	0	1

that a leaking pipe has less impact and there is still a high probability there is a low flow in the output, though it may be diminished to no flow.

The split-shaped component is modeled in a similar manner to a pipe, with the exception that this component contains multiple health and output nodes. That is, the prior distributions $P(\text{SplitHealth}_i) = P(\text{PipeHealth}_j)$, the distribution of the 'splittingIn' is determined by the output in its parent component, and

$$P(\text{SplittingOut}_i|\text{SplittingIn} = v, \text{SplitHealth}_i = w) = P(\text{PipeOut}_j|\text{PipeIn}_j = v, \text{PipeHealth}_j = w)$$

for values of i, j (instances of the named components), v and w.

4. Experiments and Evaluation

We have checked the validity of the proposed models in order to prove their usefulness. For the purpose of evaluating the diagnostic capabilities of our weighted model counting method, the 3rd aim of the research, we have generated pipe Bayesian networks 'pipes-sensors-w-h' with height h and width w as described in Section 3.2. This results in w parallel pipes that run h layers deep. Each pipe in the first layer is directly connected to one pipe in the second layer downstream, and so on, for h layers. This effectively lengthens the pipes in the first layer, and creates w parallel pipes of length h. Each pipe in every layer has a sensor that registers *flow* or *noflow*, and is identified by the coordinate in its name. For instance, 'sensor2_3' is the sensor attached to the second pipe in layer 3.

4.1. Experimental Setup

We report some experimental results of our software tool ParaGnosis and various pipe Bayesian MBD models. All below experiments ran on a system with as CPU an Intel Hexa Core i7-8750H (2.20-4.10Ghz), 9Mb cache 45W, with Kingston HyperX 16Gb (2× 8Gb) DDR4 2400Mhz RAM.

Evidence is only set on 'FlowOut', 'FlowIn', and (some of the) sensors. Only consistent evidence is considered. This means that 'sensor2_i' $\geq$ 'sensor2_j', for $i < j$. In other terms, it is impossible for an upstream pipe to have no flow and for a downstream pipe to have flow at the same time. We also only consider FlowIn $>$ FlowOut. When we say *maxflow* to *lowflow* or *maxflow* to *noflow*, this means from FlowIn = *maxflow* to FlowOut = *lowflow*, or to FlowOut = *noflow*, respectively.

Consider network 'pipes-sensors-5-2' (5 parallel pipes, 2 layers deep). Table 3 shows diagnostic results for *maxflow* to *noflow*, Table 4 for *maxflow* to *lowflow* and Table 5 for *lowflow* to *noflow*. To allow quick reading we used the following abbreviations: 'pH' stand for 'pipeHealth'; 'jH' for 'joinHealth'; 'sH' for 'splitHealth', respectively. We compute the

Table 3. Results for network 'pipes-sensors-5-2' for maxflow to noflow. Posteriors are computed for health nodes and indicate the probability of a leak. Probabilities preceded by '+' indicate the posterior's increase compared to the prior. The following abbreviations are used: 'pH' stands for 'pipeHealth'; 'jH' for 'joinHealth'; 'sH' for 'splitHealth'.

Sensors set to noflow	**Remaining sensors are set to flow**		**Remaining sensors are not set**	
	3 variables with highest posteriors	3 variables with most increased posteriors	3 variables with highest posteriors	3 variables with most increased posteriors
sensor1_1	pH1_1 (0.738)	pH1_1 (+0.514)	pH1_1 (0.740)	pH1_1 (+0.516)
sensor1_2	jH2 (0.390)	sH1 (+0.201)	sH1 (0.388)	sH1 (+0.198)
	jH3 (0.390)	pH1_2 (+0.024)	jH2 (0.380)	pH1_2 (+0.024)
sensor2_1	pH2_1 (0.738)	pH2_1 (+0.514)	pH2_1 (0.740)	pH2_1 (+0.516)
sensor2_2	jH1 (0.390)	sH2 (+0.201)	sH2 (0.388)	sH2 (+0.198)
	jH3 (0.390)	pH2_2 (+0.024)	jH1 (0.380)	pH2_2 (+0.024)
sensor3_1	pH3_1 (0.738)	pH3_1 (+0.514)	pH3_1 (0.740)	pH3_1 (+0.516)
sensor3_2	jH1 (0.390)	sH3 (+0.201)	sH3 (0.388)	sH3 (+0.198)
	jH2 (0.390)	pH3_2 (+0.024)	jH1 (0.380)	pH3_2 (+0.024)
sensor4_1	pH4_1 (0.738)	pH4_1 (+0.514)	pH4_1 (0.740)	pH4_1 (+0.516)
sensor4_2	jH1 (0.390)	sH4 (+0.201)	sH4 (0.388)	sH4 (+0.198)
	jH2 (0.390)	pH4_2 (+0.024)	jH1 (0.380)	pH4_2 (+0.024)
sensor5_1	pH5_1 (0.738)	pH5_1 (+0.514)	pH5_1 (0.740)	pH5_1 (+0.516)
sensor5_2	jH1 (0.390)	sH5 (+0.201)	sH5 (0.388)	sH5 (+0.198)
	jH2 (0.390)	pH5_2 (+0.024)	jH1 (0.380)	pH5_2 (+0.024)
sensor1_2	pH1_2 (0.579)	pH1_2 (+0.354)	pH1_1 (0.444)	pH1_1 (+0.219)
	jH2 (0.390)	sH1 (+0.039)	pH1_2 (0.444)	pH1_2 (+0.219)
	jH3 (0.390)	jH2 (+0.011)	jH2 (0.380)	sH1 (+0.103)
sensor2_2	pH2_2 (0.579)	pH2_2 (+0.354)	pH2_1 (0.444)	pH2_1 (+0.219)
	jH1 (0.390)	sH2 (+0.039)	pH2_2 (0.444)	pH2_2 (+0.219)
	jH3 (0.390)	jH1 (+0.011)	jH1 (0.380)	sH2 (+0.103)
sensor3_2	pH3_2 (0.579)	pH3_2 (+0.354)	pH3_1 (0.444)	pH3_1 (+0.219)
	jH1 (0.390)	sH3 (+0.039)	pH3_2 (0.444)	pH3_2 (+0.219)
	jH2 (0.390)	jH1 (+0.011)	jH1 (0.380)	sH3 (+0.103)
sensor4_2	pH4_2 (0.579)	pH4_2 (+0.354)	pH4_1 (0.444)	pH4_1 (+0.219)
	jH1 (0.390)	sH4 (+0.039)	pH4_2 (0.444)	pH4_2 (+0.219)
	jH2 (0.390)	jH1 (+0.011)	jH1 (0.380)	sH4 (+0.103)
sensor5_2	pH5_2 (0.579)	pH5_2 (+0.354)	pH5_1 (0.444)	pH5_1 (+0.219)
	jH1 (0.390)	sH5 (+0.039)	pH5_2 (0.444)	pH5_2 (+0.219)
	jH2 (0.390)	jH1 (+0.011)	jH1 (0.380)	sH5 (+0.103)

posteriors of value *leak* for all health variables, given the evidence that a set of sensors is set to *noflow*. This set can be found in the leftmost column. 'FlowOut' and 'FlowIn' are also observed respectively.

We compare two experiments. Columns 2-3 represent the experiment where all sensors are observed. Unobserved sensors (those not present in column 1) are set to *flow*. Columns 4-5 represent the experiment where unobserved sensors are not set. Columns 2 and 4 contain the health variables with the highest *leak* probability, whereas column 3 and 5 contain the health variables with the most increased *leak* posteriors, compared to the posteriors computed with only 'FlowOut' and 'FlowIn' in the evidence.

4.2. Observations and Diagnostic Results

The posteriors in Table 3 (*maxflow* to *noflow*) show that diagnoses are consistent with the evidence. Consider evidence 'sensor1_1' and 'sensor1_2' equal to *noflow* and all remaining sensors are set to *flow*. The most likely location for a leak is 'pipeHealth1_1'. When the evidence does not include values for sensor1_1, we see that pipeHealth1_2 indicates a leak. Resulting diagnoses, consisting of the highest probabilities for the health variables, seemed to correspond to what we expected.

When removing the *flow* sensors from the evidence we see that the evidence 'sensor1_2' does not lead to a definitive leak (a probability greater than 0.5) as indicated by pipeHealth1_2. However, pipeHealth1_2 has the highest leak probability along with upstream pipeHealth1_1. Their posteriors also have increased the most as indicated in the last column. This finding is also logical, we have not set 'sensor1_1' to *flow*, thus the leak can still be in any layer.

For Table 4 and Table 5 we see the same behavior, thereby validating the diagnostic capabilities of our Bayesian MBD approach.

4.3. Larger Networks

We have created larger systems using the description in Section 4.1, and perform weighted model counting using PARAG-

Table 4. Results for network 'pipes-sensors-5-2' for maxflow to lowflow. Posteriors are computed for health nodes and indicate the probability of a leak. Probabilities preceded by '+' indicate the posterior's increase compared to the prior. The following abbreviations are used: 'pH' stands for 'pipeHealth'; 'jH' for 'joinHealth'; 'sH' for 'splitHealth'.

Sensors set to noflow	**Remaining sensors are set to flow**		**Remaining sensors are not set**	
	3 variables with highest posteriors	3 variables with most increased posteriors	3 variables with highest posteriors	3 variables with most increased posteriors
sensor1_1	pH1_1 (0.752)	pH1_1 (+0.522)	pH1_1 (0.750)	pH1_1 (+0.520)
sensor1_2	sH1 (0.388)	sH1 (+0.201)	sH1 (0.386)	sH1 (+0.199)
	jH2 (0.264)	pH1_2 (+0.021)	pH1_2 (0.251)	pH1_2 (+0.020)
sensor2_1	pH2_1 (0.752)	pH2_1 (+0.522)	pH2_1 (0.750)	pH2_1 (+0.520)
sensor2_2	sH2 (0.388)	sH2 (+0.201)	sH2 (0.386)	sH2 (+0.199)
	jH1 (0.264)	pH2_2 (+0.021)	pH2_2 (0.251)	pH2_2 (+0.020)
sensor3_1	pH3_1 (0.752)	pH3_1 (+0.522)	pH3_1 (0.750)	pH3_1 (+0.520)
sensor3_2	sH3 (0.388)	sH3 (+0.201)	sH3 (0.386)	sH3 (+0.199)
	jH1 (0.264)	pH3_2 (+0.021)	pH3_2 (0.251)	pH3_2 (+0.020)
sensor4_1	pH4_1 (0.752)	pH4_1 (+0.522)	pH4_1 (0.750)	pH4_1 (+0.520)
sensor4_2	sH4 (0.388)	sH4 (+0.201)	sH4 (0.386)	sH4 (+0.199)
	jH1 (0.264)	pH4_2 (+0.021)	pH4_2 (0.251)	pH4_2 (+0.020)
sensor5_1	pH5_1 (0.752)	pH5_1 (+0.522)	pH5_1 (0.750)	pH5_1 (+0.520)
sensor5_2	sH5 (0.388)	sH5 (+0.201)	sH5 (0.386)	sH5 (+0.199)
	jH1 (0.264)	pH5_2 (+0.021)	pH5_2 (0.251)	pH5_2 (+0.020)
sensor1_2	pH1_2 (0.653)	pH1_2 (+0.423)	pH1_1 (0.459)	pH1_1 (+0.228)
	jH2 (0.283)	sH1 (+0.028)	pH1_2 (0.459)	pH1_2 (+0.228)
	jH3 (0.283)	jH2 (+0.015)	sH1 (0.292)	sH1 (+0.105)
sensor2_2	pH2_2 (0.653)	pH2_2 (+0.423)	pH2_1 (0.459)	pH2_1 (+0.228)
	jH1 (0.283)	sH2 (+0.028)	pH2_2 (0.459)	pH2_2 (+0.228)
	jH3 (0.283)	jH1 (+0.015)	sH2 (0.292)	sH2 (+0.105)
sensor3_2	pH3_2 (0.653)	pH3_2 (+0.423)	pH3_1 (0.459)	pH3_1 (+0.228)
	jH1 (0.283)	sH3 (+0.028)	pH3_2 (0.459)	pH3_2 (+0.228)
	jH2 (0.283)	jH1 (+0.015)	sH3 (0.292)	sH3 (+0.105)
sensor4_2	pH4_2 (0.653)	pH4_2 (+0.423)	pH4_1 (0.459)	pH4_1 (+0.228)
	jH1 (0.283)	sH4 (+0.028)	pH4_2 (0.459)	pH4_2 (+0.228)
	jH2 (0.283)	jH1 (+0.015)	sH4 (0.292)	sH4 (+0.105)
sensor5_2	pH5_2 (0.653)	pH5_2 (+0.423)	pH5_1 (0.459)	pH5_1 (+0.228)
	jH1 (0.283)	sH5 (+0.028)	pH5_2 (0.459)	pH5_2 (+0.228)
	jH2 (0.283)	jH1 (+0.015)	sH5 (0.292)	sH5 (+0.105)

NOSIS (Dal et al., 2023). It is clear that the created systems are particularly strenuous on the inference side, due to the joining node in the network. Its CPT increases exponentially when width w increases. Table 6 shows compilation and inference times of these networks. The results reflect the aforementioned comment, as compilation and inference time most notably increase as the width of the network increases. As a comparison, the well known Munin network (Jensen & Andreassen, 2008) is considered to be large, and has 1041 variables with 98423 probabilities (Dal et al., 2021). This demonstrates the inference capabilities of weighted model counting using PARAGNOSIS (Dal et al., 2023). Fig. 7 show a nearly linear increase in compilation time with respect to the number of probabilities in the networks. However, the increase in compilation and inference time is more exponential in nature as we increase the width of the network.

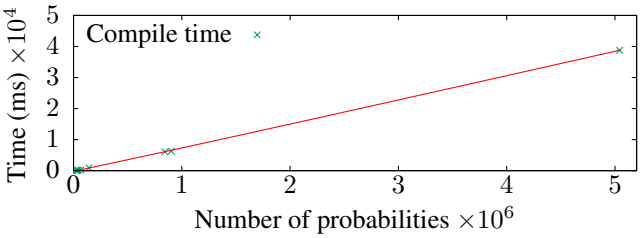


Figure 7. Compilations time with respect to the number of probabilities.

5. DISCUSSION AND CONCLUSIONS

Following the analysis of the diagnostic behavior, it appears that the compositional Bayesian-network structures developed in the sections above display behavior that is at least partly natural and intuitive. For example, a Bayesian model-based diagnostic model of a pipe system that has no input flow will raise an inconsistency in its (conditional) probability distribution if its output flow is assumed to be low or maximal. Another interesting aspect of the behavior is that faults are assumed to be closest (in terms of path length) to the entered observations that are incompatible with the expected behavior. This is a consequence of the propagation of probabilistic

Table 5. Results for network 'pipes-sensors-5-2' for lowflow to noflow. Posteriors are computed for health nodes and indicate the probability of a leaku. Probabilities preceded by '+' indicate the posterior's increase compared to the prior. The following abbreviations are used: 'pH' stands for 'pipeHealth'; 'jH' for 'joinHealth'; 'sH' for 'splitHealth'.

Sensors set to noflow	**Remaining sensors are set to flow**		**Remaining sensors are not set**	
	3 variables with highest posteriors	3 variables with most increased posteriors	3 variables with highest posteriors	3 variables with most increased posteriors
sensor1_1	sH1 (0.506)	pH1_1 (+0.293)	pH1_1 (0.508)	pH1_1 (+0.303)
sensor1_2	pH1_1 (0.498)	sH1 (+0.216)	sH1 (0.500)	sH1 (+0.210)
	jH2 (0.360)	pH1_2 (+0.042)	jH2 (0.356)	pH1_2 (+0.044)
sensor2_1	sH2 (0.506)	pH2_1 (+0.293)	pH2_1 (0.508)	pH2_1 (+0.303)
sensor2_2	pH2_1 (0.498)	sH2 (+0.216)	sH2 (0.500)	sH2 (+0.210)
	jH1 (0.360)	pH2_2 (+0.042)	jH1 (0.356)	pH2_2 (+0.044)
sensor3_1	sH3 (0.506)	pH3_1 (+0.293)	pH3_1 (0.508)	pH3_1 (+0.303)
sensor3_2	pH3_1 (0.498)	sH3 (+0.216)	sH3 (0.500)	sH3 (+0.210)
	jH1 (0.360)	pH3_2 (+0.042)	jH1 (0.356)	pH3_2 (+0.044)
sensor4_1	sH4 (0.506)	pH4_1 (+0.293)	pH4_1 (0.508)	pH4_1 (+0.303)
sensor4_2	pH4_1 (0.498)	sH4 (+0.216)	sH4 (0.500)	sH4 (+0.210)
	jH1 (0.360)	pH4_2 (+0.042)	jH1 (0.356)	pH4_2 (+0.044)
sensor5_1	sH5 (0.506)	pH5_1 (+0.293)	pH5_1 (0.508)	pH5_1 (+0.303)
sensor5_2	pH5_1 (0.498)	sH5 (+0.216)	sH5 (0.500)	sH5 (+0.210)
	jH1 (0.360)	pH5_2 (+0.042)	jH1 (0.356)	pH5_2 (+0.044)
sensor1_2	pH1_2 (0.376)	pH1_2 (+0.171)	sH1 (0.372)	pH1_1 (+0.117)
	jH2 (0.360)	jH2 (+0.001)	jH2 (0.357)	pH1_2 (+0.117)
	jH3 (0.360)	jH3 (+0.001)	jH3 (0.357)	sH1 (+0.081)
sensor2_2	pH2_2 (0.376)	pH2_2 (+0.171)	sH2 (0.372)	pH2_1 (+0.117)
	jH1 (0.360)	jH1 (+0.001)	jH1 (0.357)	pH2_2 (+0.117)
	jH3 (0.360)	jH3 (+0.001)	jH3 (0.357)	sH2 (+0.081)
sensor3_2	pH3_2 (0.376)	pH3_2 (+0.171)	sH3 (0.372)	pH3_1 (+0.117)
	jH1 (0.360)	jH1 (+0.001)	jH1 (0.357)	pH3_2 (+0.117)
	jH2 (0.360)	jH2 (+0.001)	jH2 (0.357)	sH3 (+0.081)
sensor4_2	pH4_2 (0.376)	pH4_2 (+0.171)	sH4 (0.372)	pH4_1 (+0.117)
	jH1 (0.360)	jH1 (+0.001)	jH1 (0.357)	pH4_2 (+0.117)
	jH2 (0.360)	jH2 (+0.001)	jH2 (0.357)	sH4 (+0.081)
sensor5_2	pH5_2 (0.376)	pH5_2 (+0.171)	sH5 (0.372)	pH5_1 (+0.117)
	jH1 (0.360)	jH1 (+0.001)	jH1 (0.357)	pH5_2 (+0.117)
	jH2 (0.360)	jH2 (+0.001)	jH2 (0.357)	sH5 (+0.081)

information through the network which reflects the observations combined with the associated uncertainty derived from the conditional probability tables. At first thought, one may think that this behavior is not compatible with the actual real world, as one would expect that for identical components, the likelihood of failure would also be identical independent of where the component is located in the overall system. However, as *only* by observing flow input and output and sensor data allows one to locate faults, the probabilistic reasoning provided by a Bayesian model-based diagnostic model rightfully exploits that information to the maximal extent and does indeed offer information where to look first.

In principle, the Bayesian-network network structure and its associated probabilistic parameters can be learned from observational data of a working real machine, although this will be associated with some major challenges. Where it would be straightforward to learn the prior distribution of health variables from the data, for example for 'PipeHealth', it would be harder to learn the conditional distributions

$$P(\mathrm{PipeOut}|\mathrm{PipeIn} = v, \mathrm{PipeHealth} = w),$$

for values of v, w, from data, partly because these probabilities are supposed to represent generic local probabilities, whereas in real-world systems quite a lot of the local behavior is determined by non-local behavior arising elsewhere in a system. Measurements that fully explain local behavior of components will usually not be available. Instead, available sensor data can be exploited, and basically this requires the development of new methods to answer questions of how local probability distributions can be approximated with the data from real-world systems that *can be* measured. Since parameter learning does not appear straightforward, it can be expected that structure learning will be even harder. However, here one has to keep in mind that for cyberphysical systems there often is quite some knowledge available already about its architecture and functional components, which clearly makes this problem much less challenging. Thus the positive message is that with relatively little effort a good starting point for developing a Bayesian model-based diagnostic model is within reach.

It should be mentioned that the example pipe model which

Table 6. Compilation and inference times for pipes-sensors-w-h, where compilation size is the number of arithmetic operators in the compiled representation.

Network		Number of	Number of	Compilation	Compilation	Maginalization
Width w	Height h	variables	probabilities	size	time (ms)	time (ms)
5	10	223	25833	76908	98.165	4.334
5	20	423	28033	128619	116.489	8.927
5	30	623	30233	107223	112.460	6.582
5	40	823	32433	136836	116.275	9.780
5	50	1023	34633	133992	115.188	7.974
5	100	2023	45633	202545	153.610	11.971
5	200	4023	67633	341820	235.962	20.261
6	10	267	143049	127431	905.657	9.245
7	10	311	843561	498813	6085.595	37.000
7	200	5631	902081	645177	6275.068	42.671
8	10	355	5043465	463594	38781.416	36.785

was employed in the research, has some limitations. In the first place, because of the restriction in our research to probabilistic inference in discrete Bayesian networks. It would have been more natural to use hybrid Bayesian networks to represent the behavior of a pipe system, with continuous variables for the representation of flow and sensor information and discrete for the health variables. Nevertheless, discrete variables do allow one to approximate continuous variables usually to a sufficient degree. Furthermore, in the context of flow modeling, the assumption that flow can be modeled by a directed acyclic graph may be questioned, although propagation of probabilistic information goes in both directions, in and against the direction of the arcs.

The issues mentioned above do not interfere with other conclusions concerning the probabilistic diagnostic method designed and whether exact probabilistic inference is feasible for large diagnostic problems. We have also tested large versions of our model-based diagnostic models, in order to investigate the limits of weighted model counting. We are able to perform weighted model counting in networks that are significantly larger than the largest networks in the field of Bayesian networks (Dal et al., 2023).

All in all, this research contributes to ideas on how Bayesian model-based diagnosis can be applied to large cyberphysical systems. Future research should shed light on whether the proposed probabilistic diagnostic methods can be of value in diagnosing faults in other systems than the pipe systems we studied.

REFERENCES

Bryant, R. E. (1986). Graph-based algorithms for Boolean function manipulation. *Transactions on Computers*, *100*, 677–691.

Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, *172*, 772–799.

Dal, G. H., Laarman, A., & Lucas, P. J. F. (2023). ParaGnosis: A tool for parallel knowledge compilation. In *Model Checking Software: 29th International Symposium, SPIN 2023, Paris, France, April 26–27, 2023, Proceedings* (pp. 22–37).

Dal, G. H., Laarman, A. W., Hommersom, A., & Lucas, P. J. F. (2021). A compositional approach to probabilistic knowledge compilation. *International Journal of Approximate Reasoning*, *138*, 38–66.

Dal, G. H., & Lucas, P. J. F. (2017). Weighted positive binary decision diagrams for exact probabilistic inference. *International Journal of Approximate Reasoning*, *90*, 411–432.

Dal, G. H., Michels, S., & Lucas, P. J. F. (2017). Reducing the cost of probabilistic knowledge compilation. In *Proceedings of Machine Learning Research* (Vol. 73, pp. 141–152).

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, *17*, 229–264.

Darwiche, A., & Marquis, P. (2021). On quantifying literals in Boolean logic and its applications to explainable AI. *Journal of Artificial Intelligence Research*, *72*, 285–328.

de Kleer, J. (1991). Focusing on probable diagnoses. In *Proc. of AAAI-1991* (pp. 842–848).

de Kleer, J., Mackworth, A. K., & Reiter, R. (1992). Characterizing diagnoses and systems. *Artificial Intelligence*, *56*(2-3), 197–222. doi: 10.1016/0004-3702(92)90027-u

de Kleer, J., & Williams, B. C. (1987). Diagnosing multiple faults. *Artificial Intelligence*, *32*(1), 97–130. doi: 10.1016/0004-3702(87)90063-4

Dowdeswell, B., Sinha, R., & MacDonell, S. G. (2020). Finding faults: A scoping study of fault diagnostics for Industrial Cyber–Physical Systems. *Journal of Systems and Software*, *168*, 110638. doi:

10.1016/j.jss.2020.110638

Dudek, J. M., Phan, V., & Vardi, M. Y. (2020). ADDMC: Weighted model counting with algebraic decision diagrams. In *International Conference on Artificial Intelligence* (pp. 1468–1476).

Genesereth, M., & Nilsson, N. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann.

Grievink, G. (2022). *Model-based probabilistic diagnostics of cyber-physical systems.* Bachelor Thesis, University of Twente, The Netherlands.

Jensen, K., & Andreassen, S. (2008). Generic causal probabilistic networks: A solution to a problem of transferability in medical decision support. *Computer Methods and Programs in Biomedicine*, *89*(2), 189–201. doi: 10.1016/j.cmpb.2007.10.015

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Lee, E. A. (2008). Cyber physical systems: Design challenges. In *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (pp. 363–369). doi: 10.1109/ISORC.2008.25

Lucas, P. J. F. (1996). *Structures in Diagnosis: from theory to clinical application* (Published PhD Thesis). Free University (VU) of Amsterdam, The Netherlands.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan kaufmann.

R Core Team. (2024). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org`

Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence*, *32*(1), 57–95. doi: 10.1016/0004-3702(87)90062-2

Ruijters, E., & Stoelinga, M. (2015). Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, *15-16*, 29–62. doi: 10.1016/j.cosrev.2015.03.001

Scutari, M. (2024). bnlearn – an R package for Bayesian network learning and inference [Computer software manual]. Retrieved from `https://bnlearn.com`

Srinivas, S. (1994). A probabilistic approach to hierarchical model-based diagnosis. In *Uncertainty in Artifical Intelligence* (pp. 538–545). Elsevier.