Contents

Chapter 1

A Tutorial on Bayesian Networks for System Health Management

Arthur Choi

Computer Science Department, University of California, Los Angeles

Lu Zheng

Carnegie Mellon University, Silicon Valley Campus, Moffett Field

Adnan Darwiche

Computer Science Department, University of California, Los Angeles

Ole J. Mengshoel

Carnegie Mellon University, Silicon Valley Campus, Moffett Field

1.1 Introduction

Bayesian networks have established themselves as an indispensable tool in artificial intelligence, and are being used effectively by researchers and practitioners more broadly in science and engineering [31, 8, 17]. The domain of system health management, including diagnosis, is no exception. In fact, diagnostic applications have driven much of the developments in Bayesian networks over the past few decades [1, 40, 13, 20, 36, 37, 25, 34]. In this chapter, we provide a gentle and accessible introduction to modeling and reasoning with Bayesian networks, with the domain of system health management in mind.

Formally, a Bayesian network is a type of statistical model that can compactly represent complex probability distributions. Bayesian networks are particularly well-suited to modeling systems that we need to monitor, diagnose, and make predictions about, all under the presence of uncertainty. The system under consideration may be a natural one, such as a patient, where our goal may be to diagnose a disease, given the outcomes of imperfect medical tests [1, 40, 13]. The system may be artificial, such as an aerospace vehicle, where our goal may be to isolate a component failure, given a vector of unreliable sensor readings [20, 36, 37, 25, 34]. In aerospace, these goals and algorithms are useful in developing techniques for what is often referred to as fault detection, fault isolation and recovery (FDIR). Bayesian networks can also model large-scale systems, even in settings where reasoning must be done in real-time. For example, in an online system, we may want to detect impending failures, given access to possibly noisy observations.

Our goal in this chapter is to provide the reader with a high-level perspective on Bayesian networks, with a focus on systems health management. We also hope to give a taste for the capabilities that Bayesian networks bring, in terms of modeling, reasoning, and learning in complex systems. First, in Section 1.2, we highlight briefly the problem of system health management, and introduce a small system that will be used as a running example in the following few sections. Section 1.3 provides an introduction to Bayesian networks, and Section 1.4 highlights the process by which Bayesian networks are constructed. Section 1.5 illustrates the reasoning capabilities that Bayesian networks provide, again through simple examples. Section 1.6 gives a brief overview of the types of algorithms, developed in the artificial intelligence community over the past three decades, available for reasoning in Bayesian networks. Section 1.7 does similarly for the task of learning Bayesian networks from data. Section 1.8 presents a real-world application of Bayesian networks in diagnosing Electrical Power Systems, developed for a testbed located at the NASA Ames Research Center. Finally, Section 1.9 concludes this tutorial, which also includes references to relevant books and software systems, for the interested readers.

1.2 System Health Management and Uncertainty

System health management is often needed due to uncertainty regarding a system's operation as well as in the system's environment. For example, a system might be an aerospace vehicle, and we may consider how uncertainty plays a key role during a NASA vehicle's mission. Uncertainty may be induced by: (i) the varying and uncertain environments in which vehicles operate; (ii) the unpredictability of hardware and software failures that can take place in these vehicles; (iii) noisy and ambiguous sensor systems; (iv) incomplete and uncertain knowledge of the physics of dynamic systems, both terrestrially and extra-terrestrially.

Beyond the requirement for dealing with uncertainty, there are often requirements on the ease of modeling and on real-time reasoning [25]. Ease of modeling is critical for practitioners to employ Bayesian networks in their specific domains, without becoming experts on Bayesian networks themselves. Real-time reasoning is critical for those applications that require inferences made about the system to be reported within strict time limits, so that the system is able to react to it.

In the context of ADAPT [32], an electrical power system testbed at NASA discussed in more detail in Section 1.8, we can illustrate these requirements as follows. Modeling requirements can be addressed by using procedures that take as input high-level specifications, and synthesize Bayesian networks from them. The benefit of this automated approach is that it enables the modeling of real-world systems, even by practitioners who may not be well-versed in Bayesian networks. The second requirement, for real-time reasoning, may be addressed by embedding powerful and efficient reasoning algorithms into hard, real-time systems [28]. The real-time requirement has at least two facets, namely the need for predictability and the need for fast inference. Some of the reasoning algorithms we discuss in Section 1.5, for example, the compilation to arithmetic circuits, address these concerns.

1.2.1 An Electrical Circuit Example

Consider Figure 1.1, which depicts a simple electrical system, composed of a number of components and equipped with a number of sensors. First, there are three components of interest: a battery B, a relay R, and a load L, which may be, for example, a lamp or a fan. Second, there are three sensors: a current sensor C, a voltage sensor V, and a feedback sensor F on the relay. Finally, a command may be issued to the relay R, to open or close the electrical circuit. The system components are also summarized in Table 1.1.

In this simple example, there are two nominal modes of operation, depending on the most recent command given to the relay. Given an open command, the current and voltage sensors are expected to read low and the feedback sen-

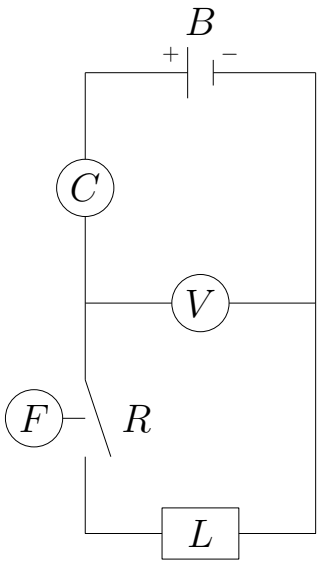


FIGURE 1.1: A diagram of a simple electrical system, with battery B, current censor (ammeter) C, voltage sensor (voltmeter) V, a relay (or switch) R, a feedback (touch) sensor F, and a load L.

TABLE 1.1: System Components

location	part	mode	health states
B	battery	healthy	healthy
		stuck disabled	stuck-disabled
		stuck enabled	stuck-enabled
W	voltage sensor	healthy	healthy
		reading stuck high	stuck-hi
		reading stuck low	stuck-lo
	current sensor	healthy	healthy
		reading stuck high	stuck-hi
		reading stuck low	stuck-lo
R	relay	healthy	healthy
		stuck open	stuck-open
		stuck closed	stuck-closed
	feedback sensor	healthy	healthy
		reading stuck open	stuck-open
		reading stuck closed	stuck-closed
L	load	healthy	healthy
		stuck disabled	stuck-disabled
		stuck enabled	stuck-enabled

sor is expected to read open. Given a close command, the current and voltage sensors are expected to read high and the feedback sensor is expected to read closed. Suppose that we have an abnormal reading, for example, if the command is to close the circuit and our current sensor reads low. We can imagine a number of faults that would explain this behavior. For example, the relay could be faulty and stuck open, or the sensor reading could be unreliable.

In this chapter, our goal is to illustrate how to reason, in an automated and principled way, about such problems by means of Bayesian networks. We shall illustrate first how to model small-scale systems such as the one described above, and then highlight how this process can be scaled to state-of-the-art approaches to real-world diagnostic problems. More specifically, we hope to illustrate that Bayesian networks provide an expressive and natural framework for modeling systems health management problems, both small and large. We further hope to illustrate that Bayesian networks provide an even more powerful framework for analyzing and reasoning about such problems. For illustrative purposes, we will use the simple electrical system of Figure 1.1 as a running example throughout the next few sections. First, we shall illustrate how a Bayesian network model can be constructed automatically from a formal system design (such as a schematic).

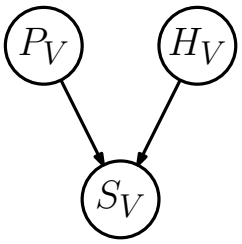
Next, we highlight some of the queries supported by Bayesian networks. For example, it is natural to ask for a given set of abnormal sensor readings:

- How likely are our observations, and how likely is each individual component to be the cause of an abnormality?
- What is the likeliest explanation of our observations, i.e., what is the likeliest joint health state?
- How robust are our estimates and explanations under perturbations in the Bayesian network?

We then provide a high-level overview of the variety of algorithms available for reasoning and learning in Bayesian networks, such as those that are more suitable for embedded systems, and powerful approximation algorithms for reasoning in large and complex systems. Finally, we highlight a successful application of the Bayesian network models and techniques that we introduce, for the purposes of diagnosing electrical power systems.

1.3 Bayesian Networks

A Bayesian network is a directed acyclic graph (DAG) that is annotated with local conditional probability tables (CPTs). Nodes represent random variables, and directed edges typically denote causal relationships between



P_V	θ_{P_V}
hi	0.5
lo	0.5

H_V	θ_{H_V}
healthy	0.99
stuck-hi	0.005
stuck-lo	0.005

P_V	H_V	S_V	θ_{S_V}
hi	healthy	read-hi	1.0
hi	healthy	read-lo	0.0
hi	stuck-hi	read-hi	1.0
hi	stuck-hi	read-lo	0.0
hi	stuck-lo	read-hi	0.0
hi	stuck-lo	read-lo	1.0
lo	healthy	read-hi	0.0
lo	healthy	read-lo	1.0
lo	stuck-hi	read-hi	1.0
lo	stuck-hi	read-lo	0.0
lo	stuck-lo	read-hi	0.0
lo	stuck-lo	read-lo	1.0

FIGURE 1.2: A Bayesian network model of a voltage sensor, along with the network's conditional probability tables (CPTs).

variables (although causality is not a requirement). For example, an edge $C \rightarrow E$ may represent a direct cause and effect relationship between random variables C and E. More precisely, the DAG of a Bayesian network encodes a set of conditional independence statements that reads as follows: Every variable is independent of its non-descendants given its parents. When edges represent direct causal influences, these independencies read: Every variable is independent of its non-effects given its direct causes.

As a concrete example, consider Figure 1.2, which depicts a simple Bayesian network model of a voltage sensor. This model is also a sub-model of a larger Bayesian network model that we shall construct for the simple electrical system in Figure 1.1 (we discuss this larger Bayesian network, in Figure 1.3, later in this chapter). In this example, we have three random variables:

- variable P_V represents the presence of voltage on a wire. The voltage may be either high, $P_V = \mathsf{hi}$, or it may be low, $P_V = \mathsf{lo}$.
- variable S_V represents the reading of a voltage sensor. The sensor either reads high, $S_V = \mathsf{read\text{-}hi}$, or it reads low, $S_V = \mathsf{read\text{-}lo}$.
- variable H_V represents the health of a sensor. We assume that a sensor has 3 modes: one nominal mode $H_V = \mathsf{healthy}$, and two failure modes, either stuck at a high reading, $H_V = \mathsf{stuck\text{-}hi}$, or stuck at a low reading, $H_V = \mathsf{stuck\text{-}lo}$.

In this example, variables P_V and H_V are parents of variable S_V, as the presence of voltage and the health of a sensor dictates the value of its reading.

In a Bayesian network, we must quantify the local relationships between a variable and its parents. In particular, we specify a CPT for each variable X in the network, which is a conditional probability distribution for a variable

X given its parents $\mathbf{U}$. These local conditional distributions induce a global probability distribution over all network variables $\mathbf{X}$. Let $\mathbf{x}$ denote an instantiation of these variables, where each variable $X \in \mathbf{X}$ is assigned to one of its values x. The probability associated with a variable instantiation $\mathbf{x}$ is then a product of these conditional probabilities $Pr(x|\mathbf{u})$ for each value of x that was set in $\mathbf{x}$:

$$Pr(\mathbf{x}) = \prod_{X \in \mathbf{X}} Pr(x|\mathbf{u})$$

where $\mathbf{u}$ are the sub-instantiations of $\mathbf{x}$ over the parent variables $\mathbf{U}$. One main feature of Bayesian networks is this compact representation of the joint probability distribution in terms of local conditional distributions.

In our example, we need to specify the conditional distribution $Pr(S_V \mid P_V, H_V)$ that a sensor has a particular reading, given the presence or absence of voltage, and the health of the sensor. In these terms, we can specify the operation of a sensor as a functional relationship: (1) if a sensor is healthy, then the sensor's reading reflects the presence or absence of voltage, or otherwise (2) if a sensor is stuck high (or stuck low), then the reading is always high (or always low). For example, we have $Pr(S_V = \mathsf{read\text{-}hi} \mid P_V = \mathsf{hi}, H_V = \mathsf{healthy}) = 1.0$, i.e., the sensor is sure to read high given that voltage is present and the sensor is healthy. We also refer to these conditional probabilities as parameters $\theta_{S_V|P_V,H_V}$ of the Bayesian network model. Variables P_V and H_V are root nodes in the DAG, so we need to specify the prior (unconditional) distributions $Pr(P_V)$ and $Pr(H_V)$, or equivalently, the parameters θ_{P_V} and θ_{H_V}. In our example, the probability $Pr(H_V = \mathsf{healthy})$ that our sensor is healthy is 99.0%, and the probabilities $Pr(H_V = \mathsf{stuck\text{-}hi})$ and $Pr(H_V = \mathsf{stuck\text{-}lo})$ that our sensor is stuck at a high and low readings are both 0.5%. We further assumed a uniform distribution for the prior $Pr(P_V)$ on the presence of voltage. We will describe in more detail in Section 1.4 the process of constructing Bayesian networks.

As we described before, the network's parameters (i.e., the three CPTs θ_{P_V}, θ_{H_V} and $\theta_{S_V|P_V,H_V}$) define a global probability distribution over the three variables P_V, H_V, and S_V which can be decomposed as follows:[1]

$$Pr(P_V, H_V, S_V) = Pr(P_V)Pr(H_V)Pr(S_V \mid P_V, H_V) = \theta_{P_V}\theta_{H_V}\theta_{S_V|P_V,H_V} \tag{1.1}$$

We can visualize this joint distribution by enumerating over all possible assignments of variables to values. Consider the following table, where we list all such assignments along with their corresponding probabilities:

[1] This decomposition follows from the chain rule, and the property that every variable is independent of its non-descendants given its parents (and thus, root variables are marginally independent).

P_V	H_V	S_V	$Pr(P_V, H_V, S_V)$
hi	healthy	read-hi	49.50% ⋆
hi	healthy	read-lo	0.00%
hi	stuck-hi	read-hi	0.25% ⋆
hi	stuck-hi	read-lo	0.00%
hi	stuck-lo	read-hi	0.00%
hi	stuck-lo	read-lo	0.25% ⋆
lo	healthy	read-hi	0.00%
lo	healthy	read-lo	49.50% ⋆
lo	stuck-hi	read-hi	0.25% ⋆
lo	stuck-hi	read-lo	0.00%
lo	stuck-lo	read-hi	0.00%
lo	stuck-lo	read-lo	0.25% ⋆

where we have starred the scenarios with non-zero probability. In this submodel, we find that the two nominal scenarios,

$$\{P_V = \mathsf{hi}, H_V = \mathsf{healthy}, S_V = \mathsf{read\text{-}hi}\} \;\&\; \{P_V = \mathsf{lo}, H_V = \mathsf{healthy}, S_V = \mathsf{read\text{-}lo}\}$$

are the most likely, both with probability 49.5%. There are four other possible scenarios (that have non-zero probability), although they are relatively unlikely scenarios that correspond to sensor failures.

In general, the number of entries in a joint probability table is exponential in the number of variables in the model. In contrast, a Bayesian network is defined in terms of local conditional distributions, i.e., its CPTs, which are relatively compact. In particular, a CPT has a size that is exponential in the number of variables that appear in it: the variable itself and its parents in the DAG. This local representation (coupled with a small number of parents for all nodes) is what allows us to compactly represent what would otherwise be an intractably large distribution.

As we shall see in Section 1.4, having to define only local CPTs greatly simplifies the process of constructing Bayesian network models. Moreover, the structure of a Bayesian network model, and the conditional independencies that we are able to infer from it, can be exploited for the purposes of efficient and effective inference and learning. In Section 1.5, we will highlight the utility of Bayesian networks in terms of the queries one can pose with respect to the joint probability distributions that they encode.

1.4 Modeling with Bayesian Networks

There are three basic approaches for building Bayesian networks [8]: (1) they can be constructed by expert knowledge, (2) they can be synthesized automatically from formal system designs, and (3) they can be learned automatically from data. Considering the domain of interest here is in system

health management, it is likely that we already have available a formal system design (such as a schematic) from which we can construct a Bayesian network, so we will focus in this section on this particular approach.

Speaking more generally, there are three basic steps for specifying a Bayesian network.

1. **Define the network variables and their values.** Certain variables represent events that we are interested in estimating, such as the health of components. Such variables are often referred to as *query* variables. Other variables represent observables or Bayesian networks inputs, such as the readings of sensors. Such variables are often referred to as *evidence* variables. A Bayesian network may also involve *intermediary* variables which are neither targets for queries, or observable events. They may aid the modeling process by relating query variables to evidence variables.

2. **Define the network structure.** That is, we need to specify the edges of a Bayesian network. In most domains, a causal interpretation of the network structure will be the most natural, where edges indicate a cause and effect relationship between variables. In this case, deciding what variables point to a variable X reduces to identifying its *direct* causes.

3. **Define the network CPTs.** How to specify a network's CPTs varies from domain to domain. In our example from the previous section, our sensor model was composed largely of functional relationships between variables, and we can rely on our domain knowledge to guide us. In other cases, we can rely on data to learn these parameters automatically, as we shall see in Section 1.7.

Consider again Table 1.1, which depicts a number of components, and Figure 1.1, which depicts how these components are connected to each other. To induce a Bayesian network model from this design, we can start by specifying a procedure for constructing Bayesian network fragments from components of interest. For example, we have already seen a sub-model for a voltage sensor, which we constructed based on the knowledge of how such sensors operate. We may incorporate this sub-model in a larger model, say by feeding the appropriate outputs of the larger model (e.g., variables that represent voltage on a wire) to the inputs of our sensor model. By connecting the sub-models of components, we can construct sub-systems, and by connecting sub-systems we can construct a complete Bayesian network which can then be used for system health management. When the system of interest is well understood and well-defined, as it typically is for electrical power systems for instance, this construction process can be completely or largely automated [24].

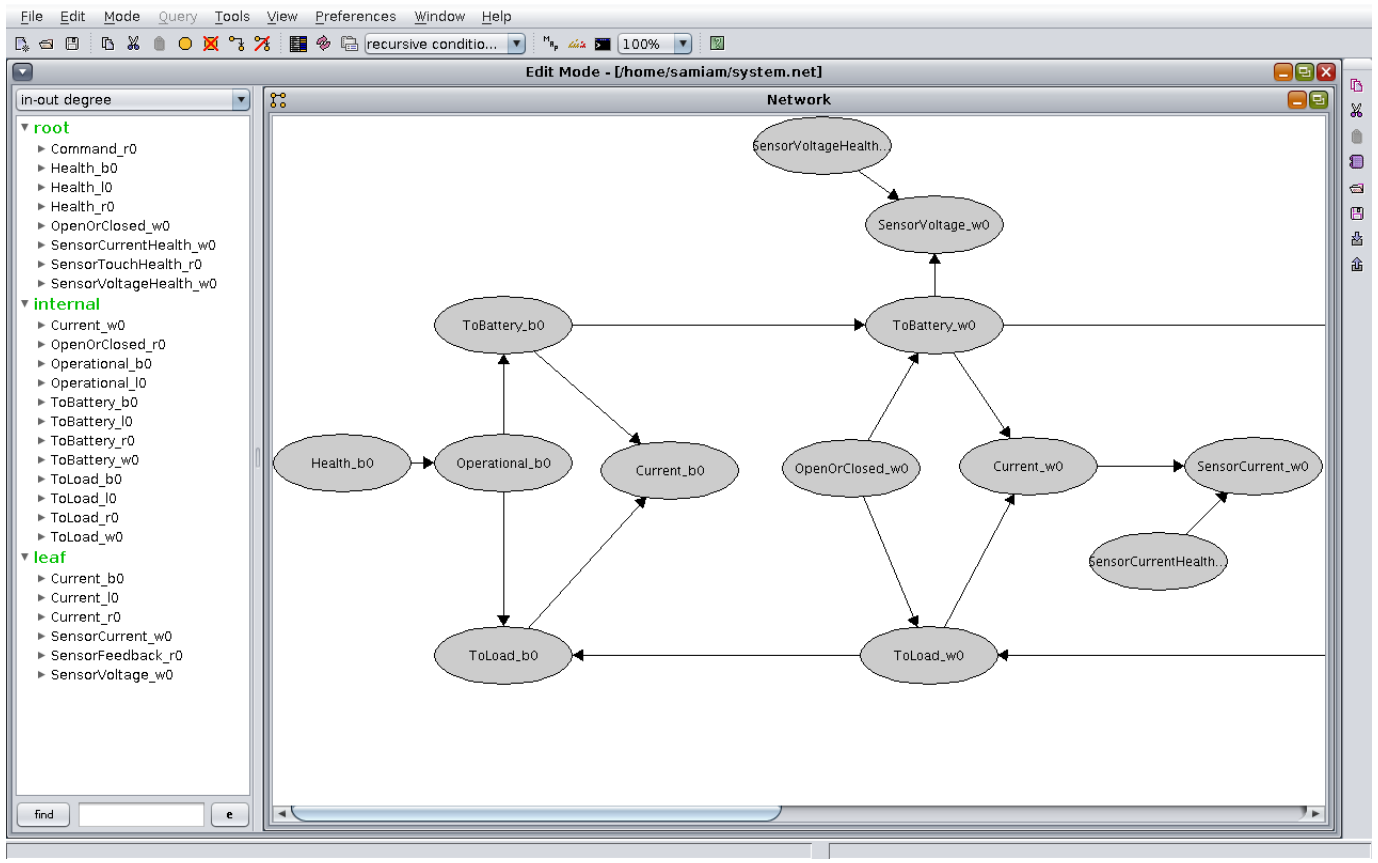


FIGURE 1.3: A partial view of a Bayesian network for the electrical system in Figure 1.1, in SamIam, a comprehensive tool for modeling and reasoning with Bayesian networks, available for download at

1.5 Reasoning with Bayesian Networks

In this section, we illustrate the reasoning capabilities enabled by a Bayesian network model. In particular, we use as an example the Bayesian network depicted in Figure 1.3, which we analyze using SamIam, a comprehensive tool for modeling and reasoning with Bayesian networks, available for download at `http://reasoning.cs.ucla.edu/samiam/`. The network used in our example is also available at `http://reasoning.cs.ucla.edu/samiam/tutorial/system.net`. Our network in this example was automatically constructed from formal design [24], where we modeled each component Table 1.1 (including the voltage sensor model from Figure 1.2), as well as their interactions as suggested by Figure 1.1.

Suppose in our electrical circuit that we issued a command for the relay to close, but after doing so, we observed unexpected sensor readings: although the current sensor reports a high current, we find that the voltage sensor reads low and the feedback sensor reads open. We denote the command by $\{R = \mathsf{close}\}$ and our sensor readings by $\{S_C = \mathsf{read\text{-}hi}, S_V = \mathsf{read\text{-}lo}, S_F = \mathsf{open}\}$. In such a situation, we are interested in inferring the failures that led to the abnormal sensor readings that we observed.

In Figure 1.3, we modeled the health of our components using variables H_B (for the battery), H_R (for the relay), and H_L (for the load). We also modeled the health of our sensors using variables H_C (for the current sensor), H_V (for the voltage sensor), and H_F (for the feedback sensor). These random variables are the focus of the queries that we will pose in our Bayesian network, with the goal of acquiring actionable health information about our electrical system.

1.5.1 Posterior Marginals

Let $\mathbf{e} = \{R = \mathsf{close}, S_C = \mathsf{read\text{-}hi}, S_V = \mathsf{read\text{-}lo}, S_F = \mathsf{open}\}$ denote the assignment of variables to values, which we call the observed *evidence*. One of the most common queries that we pose in a Bayesian network is for the posterior probability $Pr(X \mid \mathbf{e})$ of a network variable X under evidence $\mathbf{e}$.

For example, we may ask what is our belief that current is present in the circuit. In this case we have probability $Pr(P_C = \mathsf{hi} \mid \mathbf{e}) = 98.51\%$. Surprisingly, although two of our sensors suggest that the circuit is open and that there is no voltage present in the circuit, our model suggests that current is indeed flowing in our circuit. When we query for the marginal probabilities that our components are healthy, we find:

$$Pr(H_B = \mathsf{healthy} \mid \mathbf{e}) = 98.51\%$$
$$Pr(H_R = \mathsf{healthy} \mid \mathbf{e}) = 98.51\%$$
$$Pr(H_L = \mathsf{healthy} \mid \mathbf{e}) = 99.49\%$$

which is consistent with our belief that current is in fact flowing in the circuit.

Consider now the marginal probabilities of the sensor health nodes:

$$Pr(H_C = \mathsf{healthy} \mid \mathbf{e}) = 98.01\%$$
$$Pr(H_V = \mathsf{stuck\text{-}lo} \mid \mathbf{e}) = 99.01\%$$
$$Pr(H_F = \mathsf{stuck\text{-}open} \mid \mathbf{e}) = 99.01\%$$

which indicates that the voltage sensor is stuck at a low reading, and independently, that the feedback sensor is stuck at an open reading.

Note, however, that it may still be unlikely that the voltage sensor and the feedback sensor are both stuck at the same time. Indeed, observing two otherwise independent faults is typically considered much less likely than observing a single fault. As we shall soon see, observing these two faults is much more likely than any of the alternative scenarios in this particular example.

1.5.2 Explanations

In this section, we illustrate how Bayesian networks are capable of providing explanations for our observations. In our example, the Bayesian network model has suggested that current is flowing in our circuit despite two sensors suggesting otherwise. To consider why this may be, we can compute the Maximum a Posteriori (MAP) explanation.

Let $\mathbf{H} = \{H_B, H_R, H_L, H_C, H_V, H_F\}$ denote the set of our component and sensor health variables, and let $\mathbf{h}$ denote an assignment of these variables to values. Given a piece of evidence $\mathbf{e}$, the MAP explanation is then the assignment $\mathbf{h}$ having maximum probability $Pr(\mathbf{h} \mid \mathbf{e})$. In our running example, the MAP explanation is the assignment:

$$\mathbf{h} = \{H_B = \mathsf{healthy}, H_R = \mathsf{healthy}, H_L = \mathsf{healthy},$$
$$H_C = \mathsf{healthy}, H_V = \mathsf{stuck\text{-}lo}, H_F = \mathsf{stuck\text{-}open}\}$$

with probability $Pr(\mathbf{h} \mid \mathbf{e}) = 96.54\%$. According to this explanation, there are two faults: the voltage sensor is stuck at a low reading, and the feedback sensor is stuck at an open reading. This helps to explain why we observed that the probability that current is present in the circuit was 98.51%. If we look further, the *next* most likely instantiations of our variables $\mathbf{H}$ assume that there are *three* simultaneous faults, for example:

$$\mathbf{h} = \{H_B = \mathsf{disabled}, H_R = \mathsf{stuck\text{-}open}, H_L = \mathsf{healthy},$$
$$H_C = \mathsf{stuck\text{-}hi}, H_V = \mathsf{healthy}, H_F = \mathsf{healthy}\}$$

which is significantly less likely with probability $Pr(\mathbf{h} \mid \mathbf{e}) = 0.4878\%$.[2]

[2]Note if we had assumed instead that no current is flowing in the network: (1) the current sensor is stuck at a high reading, (2) we issued a close command to the relay, so either the feedback sensor is stuck or the relay is stuck, and (3) even if the circuit is open, the voltage sensor should still be able to read a voltage, so either the voltage sensor is stuck or the battery is disabled.

There are two notable classes of MAP explanations that are worth mentioning here. If the variables of interest consist of all network variables $\mathbf{X}$ (excluding those evidence variables $\mathbf{E}$ that are already observed), the MAP explanation is also called the Most Probable Explanation (MPE). In addition, the MPE is generally not as difficult to compute as MAP explanations. On the other hand, the MPE will mention any (and all) auxiliary variables that may not be of immediate interest. Moreover, the MPE and the MAP explanation may not agree on the variables of interest. On the other extreme, it is common to seek the most likely state of an individual variable X, as it can easily be identified from its marginal probabilities $Pr(X \mid \mathbf{e})$. Again, MAP explanations are in general more difficult to compute than marginals [29]. On the other hand, maximizing two variables independently may not lead to a good explanation. As we already mentioned, a Bayesian network may find that two components are likely to be faulty, yet it may still be unlikely that both are faulty at the same time.

1.5.3 Sensitivity Analysis

The question we ask now: can we quantify the degree to which we are confident in the conclusions made by a Bayesian network? Suppose, for example, that we are in the position to make a decision based on a probability computed using a Bayesian network. For example, we may decide to take one action if the probability $Pr(P_C = \mathsf{hi} \mid \mathbf{e})$ were greater than some threshold T, and another action if this probability were less than T. Can we quantify the robustness of such a decision under perturbations in the Bayesian network model?

In this case, we can appeal to sensitivity analysis where we are interested in the sensitivity of probabilistic queries with respect to changes in the network parameters of a Bayesian network. Suppose, for example, that we want to perform an action that depends on whether current is flowing in our circuit, and that we make a decision to do so if we believe that current is flowing with probability greater than 50%. In our running example, this probability was $Pr(P_C = \mathsf{hi} \mid \mathbf{e}) = 98.51\%$, which significantly surpasses our threshold. However, would a decision we make here be robust under changes in the network parameters? From a different perspective, we can ask how much do we need to change our network parameters so that $Pr(P_C = \mathsf{hi} \mid \mathbf{e}) < 50.0\%$, contradicting our current decision.

These types of queries are supported by Bayesian networks [3, 8], and in this case, one finds that significant changes in the model would be called for to contradict our current decision. For example, the parameter $\theta_{S_V=\mathsf{stuck\text{-}lo}}$, which is the prior probability that the voltage sensor is stuck on a low reading, would have to be changed from 0.5% to less than 0.0051%. This corresponds to a sensor that is around two orders of magnitude more reliable, which is a fairly extreme assumption. We find that a similar change is required in the parameter $\theta_{S_F=\mathsf{stuck\text{-}open}}$, the prior probability that the feedback sensor is stuck

open. We could also change, for example, the parameter $\theta_{S_C=\mathsf{healthy}}$, the prior probability that the current sensor is healthy. In this case, we need to change the value from 99.0% to less than 33.78% to contradiction our decision, which is also quite an extreme change.

We can also perform sensitivity analysis on the MPE, and ask what parameter changes are required to change the likeliest scenario [4]. The MPE in our running example also identifies two faults: that the voltage sensor is stuck at a low reading, and that the feedback sensor is stuck open. Again, we find that to flip the most likely scenario calls for extreme changes to our parameters. For example, we would need to change the value of the parameter $\theta_{S_C=\mathsf{healthy}}$ from 99.0% to less than 33.33%.

1.6 Reasoning Algorithms

There are several classes of algorithms for reasoning in Bayesian networks, for computing marginal probabilities and finding explanations. Each algorithm has its own advantages, which we shall highlight here.

Elimination algorithms include variable elimination [42, 10] and jointree algorithms [15, 19]. These are among the most common algorithms implemented in practice. These types of algorithms take advantage of the global structure of a Bayesian network in order to perform inference efficiently. In particular, these types of algorithms are exponential only in the *treewidth* of a Bayesian network, which is a graph theoretic property that measures, intuitively, the degree to which a graph resembles a tree [8]. Inference in a tree-structured Bayesian network, for example, can be performed efficiently in time that is only linear in the diameter of the network [31].

Conditioning algorithms include cutset conditioning [31] and recursive conditioning [9]. Intuitively, these are approaches based on decomposing the inference task into simpler cases, where each case can be solved easily, and then aggregating the results. These algorithms can have attractive computational properties compared to elimination-based algorithms. Recursive conditioning for example allows a natural anyspace-inference framework, where one is allowed to trade space with time [9].

Compilation algorithms are generally based on compiling a Bayesian network into arithmetic circuits [5, 7]. The aim of these algorithms is to push most of the computational overhead into an offline phase, while producing secondary structures (arithmetic circuits) that can be processed quite efficiently in the online inference phase. Compilation-based algorithms can be based on elimination or conditioning techniques and seek to take advantage of the global structure of a network, in addition to the *local* structure that may be present in the network CPTs. Exploiting local structure can be quite time consuming

in general, but since this is done only offline, the overhead is not incurred during the critical online phase.

All of the inference approaches discussed above lead to exact results and are therefore known as exact inference methods. In the case exact inference is not tractable, we can appeal to approximation algorithms, which have been influential in their own right in numerous applications.

"Loopy" belief propagation, along with many other message-passing algorithms inspired by it, have in the past decade been particularly successful, and have enabled new and influential applications in fields as diverse as information theory [11], computer vision, satisfiability and more broadly in artificial intelligence [31, 41]. More recent generalizations of belief propagation also provide a means by which computational accuracy can be traded for computational resources. For example, the perspective given by [6] formulates belief propagation as a way to compensate for structural relaxations, which yields a spectrum of approximation. On one end, with a coarse, fully-disconnected approximation, we have loopy belief propagation. On the other end, with the original unrelaxed model, we have exact inference. Given constraints on the computational resources available, we can then to try to identify an approximation along this spectrum that is as accurate as possible.

Stochastic sampling and search algorithms, such as importance sampling, Gibbs sampling, and stochastic local search, have the attractive property that they tend to give increasingly accurate estimates as they are given more time to run and can also work well on Bayesian networks with high treewidth. Stochastic sampling algorithms are typically used to compute marginals [12, 39], while stochastic local search algorithms are more successful when computing MAP and MPE [29, 23, 27]. The disadvantage of these algorithms is that an impractical number of samples may be needed to produce accurate estimates, and convergence may be hard to diagnose.

Variational algorithms have also been successful, particularly in applications such as information retrieval and extraction, as well as text analysis [2]. Variational approaches reduce the inference problem to an optimization task. For example, we may specify an approximation with a tractable structure, and optimize its parameterization so as to minimize the distance between the original and approximate distributions (typically the Kullback-Leibler divergence). Another attractive property of variational approaches is that they tend also to yield bounds on quantities of interest [14].

1.7 Learning Bayesian Networks

As we have discussed before, Bayesian networks can be constructed by human domain experts or they can be synthesized automatically from de-

sign. In this section, we highlight how Bayesian networks can also be learned automatically from data.

Consider again our voltage sensor model from Figure 1.2, and the following data set:

case	P_V	H_V	S_V
1	hi	healthy	read-hi
2	lo	healthy	read-lo
3	hi	healthy	read-hi
4	hi	stuck-hi	read-hi
5	lo	?	read-hi
6	?	healthy	read-lo
⋮	⋮	⋮	⋮

which consists of a number of cases where we have observed the state of a voltage sensor, which consists of three values: the presence of voltage P_V, the health of the sensor H_V, and the sensor reading S_V. In this example, we see that in some of these cases, particular values may be missing, which are marked with a "?".[3]

One can use such a data set to learn the network parameters given its structure, or learn both the structure and its parameters. Learning parameters only is an easier task computationally. Moreover, learning either structure or parameters always becomes easier when the data set is complete — that is, the value of each variable is known in each data record.

Since learning is an inductive process, one needs a principle of induction to guide the learning process. The two main principles for this purpose lead to the maximum likelihood and Bayesian approaches to learning. The maximum likelihood approach favors Bayesian networks that maximize the probability of observing the given data set. The Bayesian approach on the other hand uses the likelihood principle in addition to some prior information which encodes preferences on Bayesian networks.

We remark here that the term "Bayesian network" does not necessarily imply a commitment to the Bayesian approach for learning networks. This term was coined by Judea Pearl [30] to emphasize three aspects: the often subjective nature of the information used in constructing these networks; the reliance on Bayes conditioning when reasoning with Bayesian networks; and the ability to perform causal as well as evidential reasoning on these networks, which is a distinction originally underscored by Thomas Bayes.

The above learning approaches are meant to induce Bayesian networks that are meaningful, independently of the tasks for which they are intended. Consider for example a network which models a set of diseases and a corresponding set of symptoms. This network may be used to perform diagnostic

[3]Typically, the health of a sensor is unobservable. However, we can learn Bayesian networks with hidden or latent variables, where a variable's value will always be missing in data.

tasks, by inferring the most likely fault given a set of observed sensor readings. It may also be used for prediction tasks, where we infer the most likely sensor readings given some faults. If we concern ourselves with only one of these tasks, say diagnostics, we can use a more specialized induction principle that optimizes the diagnostic performance of the learned network. In machine learning jargon, we say that we are learning a *discriminative* model in this case, as it is often used to discriminate among failures according to a predefined set of classes (e.g., stuck or healthy). This is to be contrasted with learning a generative model, which is to be evaluated based on its ability to generate the given data set, regardless of how it performs on any particular task.

1.7.1 Learning a Bayesian Network's Parameters

Suppose we are given a data set $\mathcal{D}$ consisting of N cases $\mathcal{D}_i$, and that we have the graph structure G of a Bayesian network whose parameters we want to learn from data set $\mathcal{D}$ (the graph structure may have, for example, been given to us by an expert, or synthesized from design).

Let θ denote a parameterization of this Bayesian network, and let Pr_θ denote the probability distribution induced by graph structure G and parameters θ. Under the maximum likelihood approach to learning, the maximum likelihood parameters $\theta^\star$ that we seek are those that maximize the likelihood of the data:

$$\theta^\star = \operatorname*{argmax}_\theta \prod_{i=1}^{N} Pr_\theta(\mathcal{D}_i)$$

(assuming that the cases in our data $\mathcal{D}_i$ are independent and identically distributed). In the case our data is complete, computing the maximum likelihood parameters is a simple task. In the case our data is not complete, and has missing values, the computation is not as straightforward, but there are a number of effective algorithms that we can employ in this case.

1.7.2 Learning with Complete Data

In the case where our data set $\mathcal{D}$ is complete, i.e., it has no missing values, then the maximum likelihood parameter estimates have a simple closed form, that is based on the "empirical" probability distribution that the data set itself induces.

In the case of Bayesian learning, instead of seeking point estimates $\theta^\star$, we seek a posterior distribution (density) over network parameterizations $\rho(\theta \mid \mathcal{D})$, conditioned on the data. This posterior can in turn be used to identify point estimates (such as the mean or the mode), or one can otherwise average over all possible parameterizations. We remark that in the complete data case, we again have a simple closed form, when a suitable prior over parameters is assumed.

1.7.3 Learning with Incomplete Data

In the case our data set $\mathcal{D}$ is incomplete, and has missing values, the parameter learning task is more difficult since, in general, there is no tractably computable closed form. However, there are still effective learning algorithms in this case, such as gradient ascent, Gibbs sampling and expectation-maximization (EM). Such algorithms are often iterative, and intuitively, make inferences about the data to complete the missing values, which in turn are used to update the model parameters.

The EM algorithm, for example, is an efficient iterative procedure that searches for maximum likelihood estimates, in the presence of missing data. At a high level, each iteration of the EM algorithm consists of two steps: The E-step, and the M-step. We start with some initial guess of the network parameters. In the expectation, or E-step, we compute expected counts, based on our current estimates. In the M-step, we treat the expected counts like complete data, and compute the maximum likelihood estimates from them. We typically repeat this iterative process until it no longer improves the likelihood.

For more on learning Bayesian networks from data, including how to learn the structure of a Bayesian network, we recommend [8, 17].

1.8 Applications and Scalability Experiments

We now discuss recent, large-scale applications of Bayesian networks and arithmetic circuits in the area of electrical power systems. These studies demonstrate our successful probabilistic approach to diagnosis of an electrical power system (EPS) known as the Advanced Diagnostics and Prognostics Testbed (ADAPT) [32]. ADAPT facilitates the benchmarking of different technologies, and we have developed of a probabilistic approach, called ProDiagnose, to model-based diagnosis and applied it in this setting [22, 25, 33, 26, 16, 34, 35, 24]. In the industrial tracks of two international diagnostic competitions, which were based on ADAPT, ProDiagnose achieved the best scores in three of the four industrial tracks. In this section we discuss ProDiagnose and related work, illustrating how many of the techniques discussed earlier scale up in a real-world application.

1.8.1 Electrical Power Systems

Our society relies on electrical power in many ways. Obviously, there is the electrical power grid, which currently is under-going a major upgrading effort in the context of creating a smart grid. In addition, electrical power plays a key role in vehicles, including in cars and aerospace vehicles. Spacecraft and aircraft, for example, contain multiple sub-systems including navigation sys-

tems, power systems, and propulsion systems, and it is crucial to keep them all operational, even under adverse conditions. A real-world (hardware) EPS that is representative of EPSs in aerospace vehicles, namely the Advanced Diagnostic and Prognostic Testbed (ADAPT), has been developed and is located at the NASA Ames Research Center [32]. ADAPT includes capabilities for power storage, distribution, and consumption, and serves as a platform for development and proof-of-concept demonstrations of system health management concepts in a realistic environment. ADAPT also provides a controlled environment in which to inject failures, either through software or hardware, as well as for benchmarking of diagnostic and prognostics software in a repeatable manner.

For experimentation using real-world data, EPS scenarios were generated using ADAPT [32]. These scenarios cover component, sensor, or both component and sensor faults. A scenario is nominal or contains one, two, or three faults, and faults may be discrete or (parametric) continuous. In 2009 and 2010, international diagnostics competitions DXC-09 and DXC-10 were arranged with ADAPT as the real-world testbed (see `https://c3.ndc.nasa.gov/dashlink/projects/36/` and `https://www.phmsociety.org/competition/dxc/10` for information and data sets). In DXC-09, all inserted faults were abrupt and persisted until the end of the scenario. DXC-10, on the other hand, also included drift (non-abrupt) and intermittent (non-persistent) faults. In the following we discuss our development of BNs for ADAPT as well as the results of experiments with these BNs on nominal and fault scenarios.

1.8.2 Bayesian Network Models

Based on the ADAPT EPS, several Bayesian networks have been developed, compiled into arithmetic circuits, and used in diagnosis experiments. While several variants of this Bayesian network exist [25, 33, 34, 35, 24], we here focus mainly on the one that had the best performance in the industrial track of the diagnostic competition DXC-09. This Bayesian network has 671 discrete nodes and 789 edges; domain cardinalities range from 2 to 16 with an average of 2.86 [33, 34]. An ADAPT Bayesian network variant from 2008 contains 434 nodes and 482 edges, with node cardinalities ranging from 2 to 4 with mean 2.27 [25, 24].

Before we briefly discuss experimental results, we would like to highlight that this approach satisfies two important requirements that often arise in system health management, mentioned in Section 1.2, namely the modeling and real-time performance requirements [22, 25, 24]. To address the modeling requirement, we have developed a systematic way of representing electrical power systems as Bayesian networks, supported by an easy-to-use specification language and an auto-generation algorithm that generates Bayesian networks. The specification language and auto-generation algorithm are based on the principles laid out in Section 1.3, Section 1.4, and Section 1.5. To

address the real-time requirement, Bayesian networks are compiled into arithmetic circuits. We used the ACE system to compile an ADAPT Bayesian network into an arithmetic circuit and to evaluate that arithmetic circuit [5, 7] (see `http://reasoning.cs.ucla.edu/ace/`). Arithmetic circuits support real-time diagnosis by providing faster and more predictable inference trials. In addition, it is an exact reasoning system, with deterministic behavior.

We now briefly discuss experimental results that illustrate how our approach scales to real-world systems such as ADAPT, as well as how it compares to other approaches.

1.8.3 Experiments with Real-World Data

Using ADAPT data sets (see Section 1.8.1), experiments were used to benchmark the diagnostic quality of the ADAPT Bayesian networks and the ProDiagnose algorithm (see Section 1.8.2). Since our probabilistic models do not contain continuous random variables, experiments with continuous faults are especially challenging, and one key contribution of this work was how to handle continuous faults. Continuous data, including continuous sensor readings, were discretized before being used for clamping the appropriate discrete random variables in the ADAPT Bayesian network.

In early experiments, using 16 fault scenarios, each with one to three faults, a correct diagnosis was computed in a majority of the scenarios [25]. In fact, there was an exact match between our estimated diagnosis and the actual fault(s) inserted in 10 of the 16 scenarios, giving an accuracy of 62.5%. Even in cases where there was not exact agreement, our diagnosis was either partly correct or at the very least reasonable.

Later experiments were conducted as part of diagnostics competition DXC-09[4]. ProADAPT, an adaptation of ProDiagnose to the ADAPT setting of DXC-09's industrial track, achieved the best performance [33] in both the Tier 1 and Tier 2 tracks of DXC-09. ProADAPT had the highest-ranking scores in both the Tier 1 (among 9 international competitors) and Tier 2 (among 6 international competitors) categories. On the Tier 1 data set, ProADAPT's score was 72.8, while the second-best competitor's score was 59.85. On the more difficult Tier 2 data set, ProADAPT's score was 83.20, while the second-best competitor's score was 81.50. ProADAPT's Bayesian network and other diagnostic parameters were later further improved [34]. In DXC-10, ProADAPT was further extended to include novel feature transformation algorithms—including the use of cumulative sum (CUSUM) techniques from statistics [35]—for handling offset faults, drift faults, and intermittent faults using discrete Bayesian networks. In DXC-10, ProADAPT was declared the winner in one of the two industrial tracks.

[4] See `https://dashlink.arc.nasa.gov/topic/diagnostic-challenge-competition/` for information about the competition.

1.8.4 Experiments with Synthetic Data

In addition to experiments with real-world data from ADAPT, we have also performed experiments with synthetic data generated from ADAPT BNs [25, 24]. The purpose of these experiment was to understand the performance of arithmetic circuit evaluation (using ACE) versus alternative Bayesian network reasoning algorithms, and variable elimination (VE) and jointree propagation (JTP) in particular. Synthetic data was created by a procedure that (i) generated a set of failure scenarios according to the probabilities of the ADAPT Bayesian network's health nodes, and (ii) generated evidence by performing stochastic simulation for each failure scenario. These evidence sets were then used as evidence in the three different inference systems. Both MPEs and marginals were computed for 200 simulated evidence sets generated from an ADAPT Bayesian network.

The following timing measurements are for a PC with an Intel 4 1.83 GHz processor with 1 GB RAM and running Windows XP. Considering means only, MPE computation took on average 17.79 ms for VE and 0.2370 ms for ACE, with standard deviations of 1.513 and 0.2137 respectively. Computation of posterior marginals, on the other hand, required on average 10.02 ms for JTP and 0.6981 ms for ACE, with standard deviations of 4.451 and 0.6669 respectively [25].

In summary, we have observed strong performance for our probabilistic approach in these synthetic ADAPT experiments, and would like to make two main points. First, all three approaches are relatively fast when using this ADAPT BN, with average computation time always less than 18 ms. Second, both marginals and MPEs were computed in less than 1 ms by ACE, while the competing approaches (JTP and VE) needed 10 ms or more.

1.8.5 Discussion

Empirically, we have shown that the ProDiagnose diagnostic algorithm combined with arithmetic circuits compiled from Bayesian networks provides high-quality diagnostic results for a real-world EPS called ADAPT. We believe that these ADAPT case studies clearly demonstrate the power of our Bayesian network modeling principles and how arithmetic circuits offer a scalable inference technique with potential for real-time evaluation in aircraft and spacecraft, thereby enabling substantial improvements in aviation safety in the future. This technology has potential impact on system health management beyond EPSs in aerospace. While we discussed EPS diagnosis above, it is clear that our approach supports real-time, on-line diagnosis and prognosis for many types of large-scale systems more generally. Consequently, these empirical results should be of interest to the broader system health management community interested in fast and exact computation with predictable inference times.

1.9 Conclusion

Over the past few decades, the Bayesian network research community has developed powerful tools and algorithms for modeling, reasoning and learning complex systems. These tools and algorithms have been leveraged to solve challenging problems in fields as diverse as: artificial intelligence and machine learning; information theory [11]; computer vision; computational biology; text analysis and information extraction [2, 21]; and in diagnosis, prognosis, and system reliability [18]. The ADAPT electrical power system, which we highlighted in this chapter, is one of many examples where Bayesian networks have been applied to real-world diagnostic problems.

In this chapter, we have sought to illustrate that Bayesian networks provide a simple and natural language for modeling problems in systems health management. Moreover, as we highlighted, they support a variety of probabilistic queries, which provide a means to qualitatively and quantitatively reason about system health and reliability. Further, there are a variety of effective and principled approaches to inducing Bayesian networks from data, with or without prior expert knowledge.

There are several popular software systems for modeling and reasoning with Bayesian networks that are freely available. In this chapter, we employed UCLA's SamIam (Sensitivity Analysis, Modeling, Inference and More) system for performing the variety of probabilistic queries we considered.[5] ACE, also developed at UCLA, is an advanced system for reasoning in Bayesian networks that was successfully used in the ADAPT system described in the previous section.[6] Other popular alternatives include, Kevin Murphy's Bayesian Network Toolbox for use in Matlab computing environments,[7] and the University of Pittsburgh's GeNIe & SMILE system, which includes specialized features for Bayesian networks used in diagnostic applications.[8] There are also a variety of commercial systems for modeling and reasoning in Bayesian networks.

There are several excellent textbooks available covering Bayesian networks. A good undergraduate-level introduction is provided in Russel & Norvig's popular AI textbook [38], now in its 3rd edition. We recommend Darwiche's recent book [8], which provides an accessible introduction to modeling with Bayesian networks, as well as a thorough treatment of both classical and advanced algorithms for reasoning and learning. Another recent book by Koller & Friedman [17] provides a thorough and comprehensive treatment on probabilistic graphical models (including Bayesian networks), which we recommend for readers with a background in machine learning. We also suggest Judea Pearl's seminal book on Bayesian networks [31].

[5] Available for download at `http://reasoning.cs.ucla.edu/samiam/`
[6] Available for download at `http://reasoning.cs.ucla.edu/ace/`
[7] Available for download at `http://code.google.com/p/bnt/`
[8] Available for download at `http://genie.sis.pitt.edu/`

Bibliography

[1] S. Andreassen, M. Woldbye, B. Falck, and S.K. Andersen. MUNIN – A causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 366–372, August 1987.

[2] David M. Blei and John D. Lafferty. Topic models. In Ashok Srivastava and Mehran Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, chapter 4, pages 71–93. Chapman and Hall/CRC, 2009.

[3] Hei Chan and Adnan Darwiche. When do numbers really matter? *Journal of Artificial Intelligence Research*, 17:265–287, 2002.

[4] Hei Chan and Adnan Darwiche. On the robustness of most probable explanations. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 63–71, Arlington, Virginia, 2006. AUAI Press.

[5] M. Chavira and A. Darwiche. Compiling Bayesian networks with local structure. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1306–1312, 2005.

[6] Arthur Choi and Adnan Darwiche. An edge deletion semantics for belief propagation and its practical impact on approximation quality. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2006.

[7] A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[8] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, Cambridge, UK, 2009.

[9] Adnan Darwiche. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

[10] Rina Dechter. Bucket elimination: A unifying framework for probabilistic inference. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 211–219, 1996.

[11] Brendan J. Frey and David J. C. MacKay. A revolution: Belief propagation in graphs with cycles. In *NIPS*, pages 479–485, 1997.

[12] M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier, Amsterdam, 1988.

[13] T. S. Jaakkola and M. I. Jordan. Variational probabilistic inference and the QMR-DT database. *Journal of Artificial Intelligence Research*, 10:291–322, 1999.

[14] Tommi Jaakkola. Tutorial on variational approximation methods. In D. Saad and M. Opper, editors, *Advanced Mean Field Methods*, chapter 10, pages 129–160. MIT Press, 2001.

[15] F. V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in recursive graphical models by local computation. *Computational Statistics Quarterly*, 4:269–282, 1990.

[16] W. B. Knox and O. J. Mengshoel. Diagnosis and reconfiguration using bayesian networks: An electrical power system case study. In *Proc. of the IJCAI-09 Workshop on Self-⋆ and Autonomous Systems (SAS): Reasoning and Integration Challenges*, 2009.

[17] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[18] H. Langseth and L. Portinale. Bayesian networks in reliability. *Reliability Engineering and System Safety*, 92(1):92–108, 2007.

[19] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of Royal Statistics Society, Series B*, 50(2):157–224, 1988.

[20] U. Lerner, R. Parr, D. Koller, and G. Biswas. Bayesian fault detection and diagnosis in dynamic systems. In *Proceedings of the Seventeenth national Conference on Artificial Intelligence (AAAI-00)*, pages 531–537, 2000.

[21] Andrew McCallum. Information extraction: distilling structured data from unstructured text. *ACM Queue*, 3(9):48–57, 2005.

[22] O. J. Mengshoel. Designing resource-bounded reasoners using Bayesian networks: System health monitoring and diagnosis. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 330–337, Nashville, TN, 2007.

[23] O. J. Mengshoel. Understanding the role of noise in stochastic local search: Analysis and experiments. *Artificial Intelligence*, 172(8-9):955–990, 2008.

[24] O. J. Mengshoel, M. Chavira, K. Cascio, S. Poll, A. Darwiche, and S. Uckun. Probabilistic model-based diagnosis: An electrical power system case study. *IEEE Transactions on Systems, Man, and Cybernetics*, 40(5):874–885, 2010.

[25] O. J. Mengshoel, A. Darwiche, K. Cascio, M. Chavira, S. Poll, and S. Uckun. Diagnosing faults in electrical power systems of spacecraft and aircraft. In *Proceedings of the Twentieth Innovative Applications of Artificial Intelligence Conference (IAAI-08)*, pages 1699–1705, Chicago, IL, 2008.

[26] O. J. Mengshoel, S. Poll, and T. Kurtoglu. Developing large-scale Bayesian networks by composition: Fault diagnosis of electrical power systems in aircraft and spacecraft. In *Proc. of the IJCAI-09 Workshop on Self-⋆ and Autonomous Systems (SAS): Reasoning and Integration Challenges*, 2009.

[27] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Portfolios in stochastic local search: Efficiently computing most probable explanations in Bayesian networks. *Journal of Automated Reasoning*, 46(2):103–160, 2011.

[28] D. Musliner, J. Hendler, A. K. Agrawala, E. Durfee, J. K. Strosnider, and C. J. Paul. The challenges of real-time AI. *IEEE Computer*, 28:58–66, January 1995.

[29] J. D. Park and A. Darwiche. Complexity results and approximation strategies for MAP explanations. *Journal of Artificial Intelligence Research (JAIR)*, 21:101–133, 2004.

[30] Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *In Proceedings of the Cognitive Science Society*, pages 329–334, 1985.

[31] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[32] S. Poll, A. Patterson-Hine, J. Camisa, D. Garcia, D. Hall, C. Lee, O. J. Mengshoel, C. Neukom, D. Nishikawa, J. Ossenfort, A. Sweet, S. Yentus, I. Roychoudhury, M. Daigle, G. Biswas, and X. Koutsoukos. Advanced diagnostics and prognostics testbed. In *Proceedings of the 18th International Workshop on Principles of Diagnosis (DX-07)*, pages 178–185, Nashville, TN, 2007.

[33] B. W. Ricks and O. J. Mengshoel. The diagnostic challenge competition: Probabilistic techniques for fault diagnosis in electrical power systems. In *Proc. of the 20th International Workshop on Principles of Diagnosis (DX-09)*, Stockholm, Sweden, 2009.

[34] B. W. Ricks and O. J. Mengshoel. Methods for probabilistic fault diagnosis: An electrical power system case study. In *Proc. of Annual Conference of the PHM Society, 2009 (PHM-09)*, San Diego, CA, 2009.

[35] B. W. Ricks and O. J. Mengshoel. Diagnosing intermittent and persistent faults using static bayesian networks. In *Proc. of the 21st International Workshop on Principles of Diagnosis (DX-10)*, Portland, OR, 2010.

[36] I. Rish, M. Brodie, and S. Ma. Accuracy vs. efficiency trade-offs in probabilistic diagnosis. In *Eighteenth national conference on Artificial intelligence (AAAI-02)*, pages 560–566, Edmonton, Canada, 2002.

[37] C. Romessis and K. Mathioudakis. Bayesian network approach for gas path fault diagnosis. *Journal of engineering for gas turbines and power*, 128(1):64–72, 2006.

[38] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[39] R. Shachter and M. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Uncertainty in Artificial Intelligence 5*, pages 221–231, Amsterdam, 1990. Elsevier.

[40] M.A. Shwe, B. Middleton, D.E. Heckerman, M. Henrion, E.J. Horvitz, H.P. Lehmann, and G.F. Cooper. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base: I. The probabilistic model and inference algorithms. *Methods of Information in Medicine*, 30(4):241–255, 1991.

[41] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Understanding belief propagation and its generalizations. In Gerhard Lakemeyer and Bernhard Nebel, editors, *Exploring Artificial Intelligence in the New Millennium*, chapter 8, pages 239–269. Morgan Kaufmann, 2003.

[42] Nevin Lianwen Zhang and David Poole. Exploiting causal independence in Bayesian network inference. *Journal of Artificial Intelligence Research*, 5:301–328, 1996.