UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Bayesian Network Methods for Modeling and Reliability Assessment of Infrastructure Systems

Permalink

<https://escholarship.org/uc/item/01s3t6pg>

Author

Tien, Iris

Publication Date

2014

Peer reviewed|Thesis/dissertation

**Bayesian Network Methods for Modeling and Reliability
Assessment of Infrastructure Systems**

by

Iris Tien

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Engineering – Civil and Environmental Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Armen Der Kiureghian, Chair
Professor Arpad Horvath
Professor Alan Hubbard

Spring 2014

© Copyright 2014

Iris Tien

All Rights Reserved

ABSTRACT

Bayesian Network Methods for Modeling and Reliability Assessment of Infrastructure Systems

by

Iris Tien

Doctor of Philosophy in Civil and Environmental Engineering

University of California, Berkeley

Professor Armen Der Kiureghian, Chair

Infrastructure systems are essential for a functioning society. As these systems age, however, system reliability analyses are required to identify the critical components and make decisions regarding inspection, repair, and replacement to minimize the risk of system failure. In this study, we present novel Bayesian network (BN) methodologies for the modeling and reliability assessment of infrastructure systems. In an environment where information about a system is evolving and is oftentimes uncertain, BNs are able to both update the network when new information becomes available, and handle information probabilistically to support engineering decision making under conditions of uncertainty. One of the major limitations of the BN framework, however, is the size and complexity of the system that can be tractably modeled as a BN.

In this study, we propose a novel compression algorithm that significantly reduces the memory storage requirements for a BN model, along with an inference algorithm that performs both forward and backward inference on the compressed matrices. We also present several heuristics to improve the computational efficiency of the algorithms. Through the application of these algorithms and heuristics to example systems, we show the proposed methodology to achieve significant gains in both memory storage and computation time. Together, these algorithms enable larger systems to be modeled as BNs for system reliability analysis.

In addition, we propose a methodology based on the dynamic BN (DBN) to assess the response of a structure as it evolves through time under an excitation that is stochastic, e.g., an earthquake ground motion, based on sensor measurements that are uncertain. We look at the maximum response in particular, and derive an analytical solution for estimating the distribution of the peak response. In applying the proposed DBN framework to a multi-story shear-type building, we show the method to be robust relative to uncertainties in the structural characteristics, ground characteristics, and input motion parameters. This work informs decision making in the management of structures subject to seismic hazard and for the development and design of smart structural health monitoring systems.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation Graduate Research Fellowship, the National Science Foundation Grant No. CMMI-1130061, and the University of California Chancellor's Fellowship for Graduate Study, all of which are gratefully acknowledged.

I would also like to thank Matteo Pozzi in particular for his contributions to the dynamic Bayesian network and structural health monitoring work presented in this study.

CONTENTS

1. Introduction.....	1
1.1 Motivation.....	1
1.1.1 Infrastructure system reliability	1
1.1.2 Structural health monitoring.....	2
1.2 Outline	2
2. Background	4
2.1 Introduction.....	4
2.2 System reliability assessment	4
2.2.1 Existing methods to assess system reliability	5
2.2.1.1 Reliability block diagrams.....	5
2.2.1.2 Fault trees	5
2.2.1.3 Event trees	6
2.2.1.4 Binary decision diagrams	7
2.2.1.5 Minimum cut sets and minimum link sets.....	8
2.2.2 Bayesian networks (BNs)	8
2.2.2.1 Introduction to BNs	8
2.2.2.2 Advantages	9
2.2.2.3 Limitations.....	10
2.2.2.4 Conditional probability tables (CPTs).....	10
2.2.2.5 BNs for analyzing system reliability	11
2.2.3 BN system formulation proposed in this study	12
2.3 Structural health monitoring	13
2.3.1 Dynamic Bayesian networks (DBNs).....	13
2.3.1.1 Introduction to DBNs	13
2.3.1.2 DBNs for structural health monitoring.....	14
3. Compression algorithm	16
3.1 Introduction.....	16
3.2 Compression techniques	16

3.2.1 Background.....	16
3.2.2 Run-length encoding.....	17
3.2.3 Lempel-Ziv	17
3.3 Developed compression algorithm	18
3.4 Application to an example system	22
3.4.1 Construction of compressed CPT	23
4. Inference algorithm	25
4.1 Introduction.....	25
4.2 Existing inference algorithms	26
4.2.1 Background.....	26
4.2.2 Variable elimination	26
4.2.3 Junction tree.....	27
4.3 Proposed inference algorithm	27
4.4 Example system	32
4.4.1 Results of implementing inference algorithm	32
4.5 Dependent components.....	34
4.5.1 Algorithm for dependent components	34
5. Performance of algorithms.....	37
5.1 Introduction.....	37
5.2 Test example systems	37
5.2.1 Inference	38
5.2.2 Dependent components.....	40
5.2.3 Memory storage.....	42
5.2.4 Computation time	43
5.3 Heuristic augmentations	45
5.3.1 Heuristic for compression algorithm	45
5.3.2 Heuristic for inference algorithm	48
5.3.3 Supercomponents.....	51
5.3.3.1 Algorithm for supercomponents.....	51
5.3.3.2 Example system with supercomponents.....	53

5.4 Application: power system	54
5.4.1 Inference	56
5.4.1.1 Comparison with component importance measures	58
5.4.2 Performance of proposed algorithms	59
6. Dynamic Bayesian network for structural health monitoring.....	62
6.1 Introduction.....	62
6.2 Formulation of the problem	62
6.2.1 Background.....	62
6.2.2 Ground dynamical sub-system	63
6.2.3 Structural dynamical sub-system.....	63
6.2.4 State-space representation	64
6.2.5 Modeling the seismic excitation	64
6.2.6 System evolution	66
6.2.7 Observation equation	66
6.3 DBN representation of the system	66
6.4 Kalman filter and Kalman smoother.....	67
6.5 Distribution of maximum structural response	68
6.5.1 Analytical solution.....	68
7. Application and robustness investigation of proposed DBN framework	71
7.1 Introduction.....	71
7.2 Application.....	71
7.2.1 Estimates of the response	73
7.2.2 Distribution of maximum response	74
7.2.3 Varying measurement characteristics	78
7.2.4 Robustness to uncertainty in structural parameters	82
7.2.4.1 Uncertainty in stiffness.....	82
7.2.4.2 Uncertainty in stiffness and mass	86
7.2.4.3 Uncertainty in stiffness, mass, and damping	88
7.2.5 Robustness to uncertainty in ground parameters	90
7.2.6 Robustness to uncertainty in input motion parameters.....	93

8. Summary, Conclusions, and Future Work.....	97
8.1 Summary of major contributions	97
8.2 Future work.....	98
References.....	100

1. Introduction

1.1 Motivation

1.1.1 Infrastructure system reliability

Infrastructure systems are essential for a functioning society, from distributing the water we drink, to delivering the electricity we need, to enabling transport of people and goods from source points to destination points. Our nation's infrastructure, however, is aging and becoming increasingly unreliable with potentially severe consequences. These systems are complex, comprised of many interconnected and interdependent components, and they are subject to increasing hazards, both natural and man-made. While the ideal solution may be to replace every component in every system, the reality is that resources are limited. Given a complex infrastructure network, system reliability analyses are required to identify the critical components and make decisions regarding inspection, repair, and replacement to minimize the risk of system failure.

Further, the information that we have about these systems is often uncertain and evolving in time. With these challenges, the objective of the work presented in this study is to develop a methodology to support decision making under uncertainty in system design, management, and rehabilitation with the goal of achieving efficient resource management and improved system performance. A methodology based on a Bayesian network (BN) framework is proposed. In an environment where information about a system is evolving and is oftentimes uncertain, BNs are able to both update the network when new information becomes available, and handle information probabilistically to support engineering decision making under conditions of uncertainty.

The proposed BN methodology for system reliability assessment is described. Novel algorithms developed to enable the modeling of large, complex infrastructure systems as BNs are presented. These include a compression algorithm that significantly reduces the memory storage requirements for a BN model, and an inference algorithm that performs both forward and backward inference on the compressed matrices. Several heuristics are used to reduce computation time. Through the application of these algorithms and heuristics to example systems, we demonstrate the use of the algorithms to enable modeling of

systems of increasing size and show the proposed methodology to achieve significant gains in both memory storage and computation time.

1.1.2 Structural health monitoring

In addition to reliability assessment at the system level, we are interested in analyzing reliability at the structural level. Specifically, we are interested in monitoring the response of structures to hazards that cause dynamic effects, such as earthquake ground motions. The objective of the work presented in this part of the study is to accurately assess the structural response as it evolves through time under an excitation that is stochastic and based on sensor measurements that are uncertain. A methodology based on a dynamic Bayesian network (DBN) framework is proposed. The DBN enables probabilistic analysis of the dynamically evolving structural system. In addition, we present an analytical solution for estimating the distribution of the maximum response in particular. We also show that the proposed inference method is robust relative to uncertainties in the structural characteristics, ground characteristics, and input motion parameters. This work informs decision making in the management of structures subject to seismic hazard and for the development and design of smart structural health monitoring (SHM) systems.

1.2 Outline

The study is organized as follows:

Chapter 2 provides the background for the proposed methodologies for reliability analysis that are presented in subsequent chapters. The chapter begins with a review of existing methods for assessing system reliability. It then introduces BNs and describes the advantages and limitations of BNs. The major limitation in size and complexity of a system that can be tractably modeled as a BN is discussed. Next, DBNs are introduced. The suitability of the DBN to SHM applications is discussed, and we make a review of the literature related to the use of the DBN in SHM.

Chapter 3 presents the novel compression algorithm that has been developed to address the limitation in BN modeling of systems. The chapter begins with a review of compression techniques, particularly two canonical lossless methods that inform the development of the new algorithm. The details of the proposed compression algorithm are then presented. The chapter concludes with an application of the algorithm to an example system to illustrate its effectiveness.

Chapter 4 presents the novel inference algorithm that has been developed to perform both forward and backward inference on the BN once the network has been constructed using the compression algorithm. The chapter describes the major inference algorithms, then gives the details of the proposed methodology for inference. Similar to in Chapter 3, the proposed algorithm is applied to an example system to illustrate its effectiveness.

Chapter 5 analyzes the performance of the proposed algorithms compared to existing methods. The chapter presents the results of inference on the example system, then analyzes the algorithms in terms of both memory storage and computation time. Several heuristic augmentations developed to improve the computational efficiency of the algorithms are presented. The algorithms are then applied to a power system application to demonstrate the use of the proposed algorithms for reliability assessment of infrastructure systems. Use of the proposed method for computing measures of component importance is demonstrated.

Chapter 6 presents the DBN framework developed for monitoring the response of a structure under seismic load based on sensor measurements. The chapter begins with a brief introduction to DBN, then describes the formulation of the proposed method. An analytical solution for assessing the distribution of the maximum structural response is derived.

Chapter 7 applies the proposed framework to a multi-story shear-type building. The chapter provides results of estimating the state of the building under seismic hazard and for analyzing the distribution of the maximum structural response. The effect of uncertainties in the system, both in terms of measurement characteristics and structural parameters, is investigated. The chapter concludes with an investigation of the robustness of the proposed methodology to these uncertainties.

Chapter 8 summarizes the major contributions of this work and presents an outlook for future developments in this research area.

2. Background

2.1 Introduction

This chapter provides the background for the proposed methodologies presented in this study for reliability analysis on both a system level and a structural level. It first reviews existing methods to analyze system reliability, including reliability block diagrams, fault trees, event trees, binary decision diagrams, and minimum cut and link sets. Bayesian networks (BNs) are then introduced, and we describe the advantages and limitations of the BN framework, including the major limitation of BNs in modeling systems that is addressed in this work. Previous studies using BNs to analyze system reliability are discussed, and we describe the formulation of the system for the work presented in this study. We then introduce dynamic Bayesian networks (DBNs), an extension of the BN. The chapter concludes with a review of the use of the DBN in structural health monitoring applications.

2.2 System reliability assessment

System reliability assessment is an important endeavor, particularly in the analysis of systems that are critical to society and human life. An engineering system is comprised of multiple components. In terms of their capacity to perform, e.g., to allow a certain type of flow (traffic, water, gas) to pass through, components can have discrete or continuous states. A discrete-state component can have one, two or more states, e.g., 0%, 50%, 100% of maximum flow capacity. In practice, a continuous-state component is often approximately represented in terms of a finite number of discretized states. Similarly, systems can have discrete or continuous states. Binary-state components and systems, in particular, can be in one of two possible states, e.g., working or failed. Note that a system of binary-state components can have multiple states, i.e., component survival or failure results in a system that takes one of multiple discretized values of maximum flow capacity. The opposite is also true, i.e., a system comprised of components with multiple states can be binary. For example, components that have a distribution of flow capacity can result in a system that has a capacity above or below a threshold value that indicates the system to be in the working or failed state.

In general, the performance of individual components affects overall system performance, and individual events, e.g., component failure events, affect the state of the system as a whole. Several methods that have been developed for analyzing the reliability of these systems are briefly described below. For illustration of these methods, we consider the system shown in Figure 2.1.

Figure 2.1: Example system for illustration of reliability assessment methods

The system is comprised of 4 components $C_1, \dots, C_4$. We are interested in the connectivity of the system from the source to the sink. Let us assume that both the components and the system are binary, i.e., components can be in the working or failed state resulting in a system that is connected or not. In the following sections, we will apply each of the reliability analysis methods to this system to illustrate the different approaches.

2.2.1 Existing methods to assess system reliability

2.2.1.1 Reliability block diagrams

The reliability block diagram (RBD) is a method to graphically represent the topology of a system. A RBD consists of blocks, which represent components, and links between blocks that represent paths between components. Blocks are arranged in series or parallel configurations. If any one block in a series path fails, the entire series path fails, whereas parallel paths are redundant such that all blocks in a parallel path must fail for the parallel path to fail. Figure 2.1 is a RBD. From the diagram, we see that the system consists of two branches arranged in parallel: one made up of two components in parallel, C_1 and C_2 , the other consisting of two components in series, C_3 and C_4 .

The RBD is a useful way to show the system components and their relationships. From the configuration of the system and given the reliability of the individual blocks comprising the RBD, system reliability measures can be computed. The RBD method, however, is most frequently applied to the analysis of simple systems comprised of few components (Pages and Gondran 1986, Modarres et al. 2010). Thus, RBDs are not sufficient for the reliability analysis of complex civil infrastructure systems.

2.2.1.2 Fault trees

A fault tree (FT) is a top down analysis method whereby an undesired top event, e.g., system failure, is analyzed based on the combination of lower-level events causing that failure (Rausand and Hoyland 2004). The FT is a graphical model that consists of the top

event with branches of events leading to this top event constructed below it. Events are separated by “and” or “or” gates, where an “and” gate indicates that all input events to the gate must occur for the output event to occur, and an “or” gate indicates that the output event occurs if any of the input events occurs. Analysis of system reliability is performed by analyzing the probabilities of the sequences of events that lead to the occurrence of the top event. Figure 2.2 shows the FT for the system shown in Figure 2.1, where fault is defined as loss of connectivity between source and sink.

Figure 2.2: Fault tree for example system shown in Figure 2.1

FTs have been widely used in the nuclear industry (Vesely et al., NRC 1981). However, because a FT is constructed for a particular undesired system outcome, a single FT cannot model all possible modes or causes of system failure (Biolini 2004). While the results of one FT can be fed into another FT to model subsequent failures, for complex systems, the number of branches can increase quickly. In addition, analysis using FTs is limited to system failure paths. FTs are therefore not well suited for the analysis of events that are not necessarily sequential in nature.

2.2.1.3 Event trees

In contrast to a FT, which is a deductive process and traces backwards through a cause chain given a predefined undesired top event, an event tree is an inductive method and traces forward through a causal chain to assess the risk of occurrence of different system outcomes. Figure 2.3 shows an event tree for the system shown in Figure 2.1. Because event trees model sequences of events, the event tree in Figure 2.3 assumes failure events are chronological in nature. We assume, for example, that failure of component C_1 is the initiating event. The possible outcomes of the system are a connected or a disconnected system. One, of course, will need to consider all possible initiating events.

Figure 2.3: Event tree for example system shown in Figure 2.1

Like the FT, the event tree is a graphical model. It consists of an initiating event with branches following that describe sequences of possible subsequent events. It is difficult, however, to model events of interest that are not necessarily sequential in nature. In addition, the size of an event tree can grow rapidly for complex systems. In general, the number of branches in the event tree grows exponentially with the number of sequential events (Rausand and Hoyland 2004).

2.2.1.4 Binary decision diagrams

Binary decision diagrams (BDDs) are used for binary systems in particular to model Boolean functions (Akers 1978, Bryant 1986), where a binary system is defined as comprised of binary components with a binary system variable. A BDD is a graphical model that is comprised of nodes and edges. Each node in the graph can take one of two values. The graph begins with a root node, and based on its binary value, e.g., 0 or 1, branches into the following node, which, based on its binary value branches into the subsequent node, etc. The nodes represent the variables in the Boolean function. Figure 2.4 shows the BDD for the system shown in Figure 2.1. A value of 0 indicates failure and 1 indicates survival. By convention, dotted and solid arrows indicate failure and survival paths, respectively. Components are denoted by circles, and the final values of the system are given in the squares.

Figure 2.4: Binary decision diagram for example system shown in Figure 2.1

In general, the number of nodes (and number of paths) in a BDD is exponential with the number of variables in the domain of the Boolean function (Nielsen 2000). For a system comprised of components, this corresponds with a BDD of exponentially increasing size as the number of components in the system increases, representing a significant computational challenge. A method utilizing BDDs to perform inference in BNs has been proposed. This will be discussed further in Section 2.2.2.5.

2.2.1.5 Minimum cut sets and minimum link sets

In analyzing the reliability of a coherent binary system, i.e., a system where the change in the state of a component from fail to working (from working to fail) does not decrease (increase) the system failure probability, the components of the system can be identified as belonging to minimum cut sets (MCSs) and minimum links sets (MLSs). A MCS is a minimum set of components whose joint failure indicates failure of the system. If any one MCS fails, the system fails. A MLS is its converse, i.e., a minimum set of components whose joint survival indicates survival of the system. If any one MLS survives, the system survives. For the system shown in Figure 2.1, the set of MCSs is $\{(C_1, C_2, C_3), (C_1, C_2, C_4)\}$, and the set of MLSs is $\{(C_1), (C_2), (C_3, C_4)\}$.

Several methods have been developed for the identification of MCSs/MLSs. These include enumerating MCSs one by one using recursive methods (Benaddy and Wakrim 2012), including the recursive decomposition algorithm described in Li et al. (2007). Suh and Chang (2000) propose a graphical approach using the connection matrix to obtain the MCSs. Other methods use a blocking mechanism to ensure every MCS is generated only once (Shin and Koh 1998). Methods for enumerating cut sets for k -out-of- n networks, i.e., at least k input flows out of n input edges are required to generate output flows, are proposed in Tan (2003) and Yeh (2006), with the latter proposing a depth first search algorithm to search the tree and determine the MCSs. MCSs can also be generated directly from FTs, e.g., using the MOCUS (method for obtaining cut sets) algorithm (Fard 1997, Rausand and Hoyland 2004).

2.2.2 Bayesian networks (BNs)

2.2.2.1 Introduction to BNs

A Bayesian network (BN) is a directed acyclic graph comprised of nodes and links, as shown in the example in Figure 2.5. Each node represents a random variable and each link the probabilistic dependency between the variables (Jensen and Nielsen 2007). In constructing a BN, each node in the graph is described by a set of mutually exclusive and collectively exhaustive states. For discrete random variables, the description of these states is straightforward. For continuous random variables, methods have been developed to perform exact inference in continuous linear Gaussian BNs (Lauritzen and Jensen 2001, Madsen 2012), where inference is the process of making conclusions about uncertain quantities in the system based on data or observations. For non-Gaussian variables, simulation methods such as Markov-chain Monte Carlo can be used (Neal 1993). Otherwise, continuous random variables can be discretized.

Figure 2.5: An example of a BN

Figure 2.5 shows a BN model for five random variables $\mathbf{X} = \{X_1, \dots, X_5\}$. Looking at X_4 , we see arrows into X_4 from X_1 and X_2 . In BN terminology, X_4 is called a “child” node of X_1 and X_2 , and X_1 and X_2 are called the “parent” nodes of X_4 . Children nodes are defined conditionally on the parent nodes. Specifically, for discrete nodes, a conditional probability table (CPT) is attached at each node that describes the conditional probability distribution of the node given the states of its parents. Nodes without parents, i.e., X_1 and X_2 , are known as root nodes, and are defined by their marginal probability distributions. Together, the conditional and marginal probability distributions define the joint distribution over all random variables $\mathbf{X}$ in the network according to the rule

$$p(\mathbf{x}) = \prod_{i=1}^n p(x_i | Pa(x_i)) \quad (2.1)$$

where $p(\mathbf{x}) = \Pr(X_1 = x_1 \cap X_2 = x_2 \cap \dots)$ is the joint probability mass function of the random variables, $p(x|y) = \Pr(X = x | Y = y)$ denotes the conditional probability mass function of random variable X given the outcome $Y = y$ of another random variable, and $Pa(x_i)$ indicate the parent nodes of x_i for $i = 1, \dots, n$ total nodes in the network.

2.2.2.2 Advantages

As a probabilistic framework, the BN it is well suited for modeling and analyzing the reliability of systems. Relationships between variables, representing component states, are modeled probabilistically and the likelihood that a node is in a particular state is measured in degrees of belief. This is particularly useful in an environment where information is uncertain, including in the analysis of infrastructure systems, where information is oftentimes scarce, incomplete, or probabilistic in nature. Furthermore, the BN allows a transparent framework for modeling, which facilitates model review and verification by disciplinary experts who may not be familiar with probabilistic methods. Moreover, by extending the BN with decision and utility nodes, one can construct an influence diagram, which allows decision-making in accordance with the maximum utility criterion (Bensi et al. 2011). Thus, the BN is a powerful tool for modeling and reliability analysis to support engineering decision making for infrastructure systems under conditions of uncertainty.

In addition, the BN framework allows for updating of the probability distributions as new information, or evidence, becomes available. When evidence, e.g., the information that a node is in a particular state, on one or more variables is entered into the BN, the information propagates through the network to yield updated probabilities in light of the new observations. This capability of updating is particularly advantageous when the information on which the analysis of a system is based is evolving. In infrastructure reliability assessment, this enables decision making based on the most up-to-date information, e.g., after an action is taken to repair or replace a component of the system, or a particular component failure is observed. It also enables system planning based on different scenarios, e.g., after a natural hazard event such as an earthquake has occurred as in Bensi et al. (2011).

2.2.2.3 Limitations

One of the major limitations of the BN framework, however, is the size and complexity of the system that can be tractably modeled as a BN. In particular, as described in the following section, the CPTs can grow enormously large when a node has many parents, each having many states. This happens in systems with many components. Thus, despite its advantages, BNs have been limited in their use for system reliability analysis. Previous studies of the use of BNs for modeling system performance are discussed in Section 2.2.2.5.

2.2.2.4 Conditional probability tables (CPTs)

As mentioned earlier, each node in the BN is assigned a conditional probability table, CPT, which defines the conditional probability distribution of the variable represented by the node for given states of its parents. Specifically, in the case of a discrete node, the CPT provides the conditional probability mass function of the node, given each of the mutually exclusive combinations of the states of its parent nodes. As an example, for the BN given in Figure 2.5, Table 2.1 gives the CPT of X_4 given the states of its parents X_1 and X_2 . Here, we are considering a binary system such that the nodes X_1 , X_2 , and X_4 each have two possible states, 0 or 1.

Table 2.1: Conditional probability table for X_4 from BN of Figure 2.5

X_1	X_2	X_4	
		0	1
0	0	$P(X_4=0 X_1=0, X_2=0)$	$P(X_4=1 X_1=0, X_2=0)$
	1	$P(X_4=0 X_1=0, X_2=1)$	$P(X_4=1 X_1=0, X_2=1)$
1	0	$P(X_4=0 X_1=1, X_2=0)$	$P(X_4=1 X_1=1, X_2=0)$
	1	$P(X_4=0 X_1=1, X_2=1)$	$P(X_4=1 X_1=1, X_2=1)$

The size of a CPT grows exponentially with the number of parents of the node in the graph. If each node has m states, then the CPT consists of m^{n+1} elements, where n is the number of parents. As shown in Table 2.1, the CPT of the binary node X_4 , which has two binary parents, consists of $2^3 = 8$ elements.

In modeling system reliability, the states of components that comprise the system are represented as the nodes of the graph. The model includes a node representing the system state. Because the state of the system depends on the states of its components, one simple way to model the BN is to define the system state as a child of the nodes representing the component states, i.e., links are directed from component nodes to the system node, as shown in Figure 2.6.

Figure 2.6: BN of a system comprised of n components

Thus, for a binary system comprised of $n = 100$ total components, for example, the CPT for the system node consists of $2^{101} = 2.5 \times 10^{30}$ individual elements. This poses a significant memory storage challenge in constructing and analyzing the BN. The exponential increase with each additional parent node quickly renders the problem intractable. Of course other, more efficient ways of modeling a system by BN are available, as described below and in greater detail in Section 2.2.2.5. However, all these models face serious problems of memory demand. For this reason, BN modeling of systems has been limited to small systems, typically of no more than 20 binary components. The methodologies presented in this study attempt to address this problem of exponentially increasing memory storage demand with increasing system size to enable large systems to be modeled as BNs for reliability assessment.

2.2.2.5 BNs for analyzing system reliability

Examples in the existing literature of the use of BNs for modeling system performance are limited (Bensi et al. 2011). Previous studies have focused on generating BNs from conventional system modeling methods, such as reliability block diagrams (Torres-Toledano and Succar 1998) and fault trees (Bobbio et al. 2001). However, the systems studied have been small, comprised of 5 and 10 components, respectively. It has also been proposed to map reliability block diagrams with general gates to equivalent BN models (Kim 2011). However, the method is applied to a simple 5-node system and the development of algorithms to apply to the analysis of complex systems is noted as future work and remains unaddressed.

Previous studies of the use of BNs for structural system reliability assessment have demonstrated the ability of a BN to use system-level test data to update information at the component level (Mahadevan 2001). The authors note, though, that computational effort increases significantly with an increasing number of components in the system. An approach characterized as branch and bound, whereby events of relatively low probabilities are ignored and set to zero in the construction of the CPTs for the BN, is therefore used to apply the BN framework to larger systems. The example given, however, is for a system consisting of only 8 nodes, and the willful discarding of available information, leading to a subsequent loss of accuracy in the result, is not ideal.

More recently, BNs have been used to model the reliability of slightly larger systems, including a system of 16 components in Boudali and Dugan (2005). However, the authors state that this “large number” of components makes it “practically impossible” to solve the network without resorting to simplifying assumptions or approximations. In addition, it is clear that even a network of 16 components is not enough to create a full model of many real-world systems.

A method utilizing Reduced Ordered Binary Decision Diagrams (ROBDDs) to efficiently perform inference in BNs representing large systems with binary nodes is proposed in Nielsen et al. (2000). However, a troubleshooting model is considered, which includes a major assumption of single-fault failures, i.e., exactly one component is malfunctioning, causing the system to be at fault. In general, the number of paths in the ROBDD is exponentially increasing with the number of components. It is the single-fault assumption that bounds the size of the ROBDD and enables modeling of larger systems. For more general systems, including infrastructure systems, this single-fault assumption cannot be guaranteed. Therefore, the gains from using the ROBDD may not be applicable.

Finally, a topology optimization algorithm is proposed in Bensi et al. (2013) to address the inefficiency of a converging BN structure as shown in Figure 2.6. The authors develop a methodology to create a chain-like BN model of the system based on survival or failure path sequences. This formulation minimizes clique sizes to more efficiently model large systems as BNs. The proposed optimization program, however, must consider the permutation of all component indices and, therefore, may itself become intractably large for large systems.

2.2.3 BN system formulation proposed in this study

As described in the previous section, the component states of a system directly impact the system state. Thus, we arrive at the BN formulation of the system for this study as shown in Figure 2.6. Consider a binary system, where each component as well as the system are in one of two possible states, survival or failure. Observe that the component states deterministically define the system state, i.e., for any combination of the component states, the system is in either a survival or a failure state. Such a model is particularly useful for studying the reliability of infrastructure systems, such as gas, power, or transportation systems, where the states of individual gas pipelines, electrical transmission lines, or roadways directly determine the state of the infrastructure system. While the methodolo-

gies developed in this study focus on binary systems, they can be extended to multi-state flow systems, e.g., where the component states are discretized values of a flow capacity, e.g., 0%, 50%, and 100% of maximum capacity. In such a system, the flow capacity of individual components determines the flow capacity of the overall system. In any case, for any specified states of components states, the state of the system is deterministically known.

When the component states deterministically define the system state, the CPT associated with the system node, which we have seen exponentially increases in size with the size of the system, has a special property. For the binary system, let the failure and survival states of the system be defined as 0 and 1, respectively. Since for each distinct combination of component states the system state is known with certainty, the system CPT is comprised solely of 0s and 1s, with the values given in a single vector as shown in Chapter 3. For multi-state flow systems with states $s_1, \dots, s_m$, the CPT for the system node gives the probability of the system being in a given state s_i given the states of the components. As before, the system state is known with certainty for each distinct combination of component states. Thus, for a system comprised of n components with M states each, and a system that has m possible states, the system CPT is still comprised solely of 0s and 1s, now in a matrix of size $M^n \times m$.

For both binary and multi-state systems, therefore, there are a limited number of values that enter the system CPT. Thus, while the size of the CPT is exponentially increasing with the number of components in the system, the values in the CPT are finite, containing only values of 0 or 1 in the initial construction of the BN. The novel algorithms we have developed take advantage of this property. In the following chapters, we present these algorithms and show how their use enables large, complex infrastructure systems to be modeled as BNs for system reliability analysis.

2.3 Structural health monitoring

Structural health monitoring (SHM) is the process of using observations of a structure to characterize and assess the state of the structure over time. These observations generally take the form of measurements from sensors, which are able to sample over a range of variables, including acceleration, strain, temperature, pH, corrosion, acoustic properties, etc. As sensors continue to decrease in price and increase in capability, and sensor monitoring becomes increasingly ubiquitous, SHM methods have been developed to both improve the data collected through the development of new sensor devices, as well as facilitate how this data is used to learn about the system. The focus of this study is on the latter, in how we process sensor data to perform inference on a structural system.

2.3.1 Dynamic Bayesian networks (DBNs)

2.3.1.1 Introduction to DBNs

The dynamic Bayesian network (DBN) is a probabilistic framework that models the evolution of a system or process over time. It consists of a sequence of connected BNs, each representing the system at a time slice t (Dean and Kanazawa 1989). The evolution in time is represented by directed links between nodes of successive time-slice BNs that carry information on temporal dependencies of the respective processes.

As an example, Figure 2.7 shows a DBN for a system involving four random processes, $X_1(t)$ to $X_4(t)$. Node X_i^t in the DBN represents the value of random process $X_i(t)$ at time slice t . Links directed from nodes X_1^t and X_2^t towards node X_3^t indicate that the probability distribution of the latter node is defined conditional on the former nodes. Likewise, the directed link from node X_3^t towards node X_4^t indicates that the latter node is defined conditional on node X_3^t . Finally, directed links between nodes X_3^t, X_3^{t+1} , etc., for successive time slices indicate the temporal dependence of the random process $X_3(t)$. The dependence between successive time values of the remaining nodes depends on our state of knowledge about random variables X_3^t, X_3^{t+1} , etc.

Figure 2.7: Example of a DBN to model a dynamically evolving system

2.3.2.2 DBNs for structural health monitoring

As the DBN performs probabilistic analysis of a dynamic system, it is ideal for SHM applications, where there is uncertainty in the sensor measurements and the structure is dynamically evolving in time. We are interested in performing inference on the dynamically evolving response of a structure when it is subjected to a stochastic excitation, e.g., an earthquake, based on information from sensor measurements.

Several studies have used Bayesian methods in SHM to perform damage detection in a structure. Vanik and Beck (1997) propose a Bayesian approach to account for uncertainties in the structural system being analyzed to determine the existence and location of damage. Vanik et al. (2000) use the proposed approach to continually update the stiffness parameters of a structure. A high likelihood of reduction in stiffness at a particular location is then used as a proxy for damage at that location. Sohn and Law (1997) use Bayesian methods to identify multiple damage locations using estimated modal parameters. The Bayesian framework accounts for measurement noise and modeling error between the structural and the analytical model to detect the most likely locations and

amount of damage in a structure. Yang et al. (2006) use vibration data in particular to identify changes in structural parameters and detect damage.

In addition, Bayesian techniques have been used in SHM to identify the modal parameters of a structure. A Bayesian framework is proposed in Yuen and Katafygiotis (2002) to explicitly treat uncertainties in sensor measurements and modeling assumptions to obtain distributions of the modal parameters, including the most probable values of the parameters and their uncertainties. Both Katafygiotis and Yuen (2001) and Au et al. (2013) use data from ambient vibrations for modal identification to obtain updated distributions of the modal parameters. The system identification and damage detection problems described previously that are addressed in these studies are long-term monitoring problems, however. Our interest is in performing inference on the system state as the structure is subjected to a stochastic hazard.

Previous studies on the use of Bayesian methods in the analysis of structures subject to hazards have focused on damage detection after the extreme event rather than modeling the structure during the event. Yang et al. (2007) use an extended Kalman filter approach to identify the structural parameters, such as stiffness and damping, after an unknown external excitation. The Kalman filter will be discussed in detail as the approach to processing the information in the DBN in Section 6.4. Methods to analyze non-linear post-damage behavior are discussed in Ghanem and Ferro (2005).

For the monitoring of structures during extreme events, SHM systems are proposed in Straser and Kiremidjian (1996) and Kottapalli et al. (2003). These studies are hardware system focused, however, rather than in performing probabilistic analysis of the data collected using these systems. Wu and Beck (2012) do use a Bayesian framework and expand their analysis to the monitoring of systems both before and after an earthquake. The pre-event prognosis includes probabilistic predictions of the damage to a structure due to an earthquake, while the post-event diagnosis updates these predictions using Bayesian methods. The response of the structure during the seismic event itself is not analyzed, however.

Finally, in these studies using Bayesian methods for SHM, the authors limit the use of Bayes to the standard Bayesian updating of system parameters. To our knowledge, there have been no studies using the DBN for SHM. In Chapters 6 and 7, we present our proposed DBN framework for this purpose. The objective of the framework is to be able to use uncertain sensor measurements to probabilistically analyze the response of a structure due to seismic excitation as it dynamically evolves in time.

3. Compression algorithm

3.1 Introduction

Data compression is the process of reducing the number of bits required to store a given dataset. Using the rules and methods of a particular compression algorithm, the information in the original dataset is encoded in a manner that reduces the required memory storage, though typically with increased computational costs according to the classic space-time trade-off as described in Dechter (1999). Different compression algorithms applied to a dataset may result in different degrees of reduction.

This chapter first describes two canonical compression techniques, run-length encoding and the Lempel-Ziv algorithm, then describes the proposed compression algorithm for compressing the system conditional probability table (CPT) of the Bayesian network (BN) model for system performance. The algorithm is then applied to an example system to illustrate its effectiveness.

3.2 Compression techniques

3.2.1 Background

Several methods have been developed for compressing data, including both lossy and lossless methods. In lossy compression, nonessential information is removed from the dataset to reduce storage requirements. For example, in lossy audio compression, elements of the audio signal outside the range of human hearing are removed. What is defined as nonessential can vary, and there is a trade-off between the amount of information that is preserved and the amount a dataset can be reduced. Of course, the more information that is removed, the smaller the size of the resulting dataset. However, the removal of too much information results in distortions. In all lossy compression, the compressed dataset is an approximation of the original dataset.

In lossless compression methods, redundancy in a dataset is exploited to reduce the number of bits required to store the data without loss of any information. Therefore, a dataset compressed using lossless compression can be fully reconstructed, i.e., decompressed

completely accurately. The two compression techniques described in the following sections, run-length encoding and Lempel Ziv, are both lossless.

3.2.2 Run-length encoding

In a set of data to be compressed, a run is defined as consecutive bits of the same value. In run-length encoding, runs are stored as a data value and count (Hauck 1986). For example, in compressing a sequence of white and black pixels, respectively indicated by “W” and “B,” a sequence of 18 W’s is stored as the count 18 and the data value W. Thus, the memory storage requirement is reduced from 18 elements to 2. Run-length encoding is well suited for data with many repeated values. However, mixed values are stored literally, which results in little gain for mixed data. For example, alternating white and black pixels results in the compressed dataset 1W1B1W1B ..., which in fact doubles the memory storage requirement compared to the uncompressed form WBWB An example of run-length encoding is shown below:

Uncompressed dataset	WWWWWWWBWWWWBWWWWWWWWWW
Compressed dataset	7W1B5W2B10W

In the dataset, the initial run of 7 W’s is stored as 7W, followed by one B stored as 1B, followed by a run of 5 W’s stored as 5W, etc. Thus, the original dataset comprised of 25 elements is compressed to a dataset of 10 total elements. Of course, gains achieved by employing run-length encoding to compress a given dataset depend on the number and length of runs in the dataset.

3.2.3 Lempel-Ziv

Techniques based on the classical Lempel-Ziv algorithm (Ziv and Lempel 1977) find patterns in the data, construct a dictionary of phrases, and encode based on repeated instances of phrases in the dictionary. While there are several techniques developed to construct the dictionary, the major advantage of the Lempel-Ziv algorithm lies in the ability to call the phrases in the dictionary at each repeated instance of a phrase, while having to store just one instance of the phrase in the dictionary. An example of the Lempel-Ziv algorithm with the corresponding dictionary is shown below:

Uncompressed dataset	WWBWBBWBWWBB
Compressed dataset	∅W1B2B2W3

Dictionary:

Phrase number	1	2	3	4
Full phrase	W	WB	WBB	WBW
Encoded as	∅W	1B	2B	2W

In constructing the dictionary, we begin with the empty set ∅. The dictionary is then dynamically constructed as we process through the data. Moving through the uncompressed dataset, bit by bit, we read the input. We find the longest string in the dictionary that

matches the current input. We read the phrase number as part of the compressed output and add the bit that follows. If not already present in the dictionary, the phrase is added to the dictionary as a new entry with an assigned number.

In the above dataset, our first bit is W, which we encode as the empty set $\emptyset$ plus a W. We then encounter another W, which we have seen before as phrase 1, and now append to that phrase the bit that follows, B. Thus, we create phrase 2 as phrase 1 (W) plus a B and add that to the dictionary. We then encounter another W; however, this is not the longest string in the dictionary that matches the current input. Instead, it is WB, which we have encountered as phrase 2. Therefore, we encode these bits as phrase 2 (WB) plus the B that follows and add this to the dictionary as phrase 3. In a similar way, we next encounter the phrase WBW, which we encode as phrase 2 (WB) plus a W and add to the dictionary as phrase 4. Finally, we encounter the phrase WBB, which we encode as the phrase 3 that we have previously created. Thus, the end result of the compressed dataset is $\emptyset W1B2B2W3$.

The key idea in Lempel-Ziv is to find patterns in the dataset to be compressed and to utilize the repetitions of those patterns to achieve savings in memory storage demand. In the example given, the relatively short length of the dataset resulted in limited savings in the compression. However, as the size of the dataset gets larger and the number of repeated phrases increases, the gains achieved by employing the Lempel-Ziv algorithm to compress the dataset also increase.

3.3 Developed compression algorithm

The developed compression algorithm integrates both run-length encoding and Lempel-Ziv concepts to compress the system CPT of the BN. Both run-length encoding and Lempel-Ziv result in lossless compression. Therefore, no approximations or assumptions are made in the compression of the system CPT.

As mentioned earlier, in the developed compression algorithm a run is defined as consecutive bits of the same value. In the binary case, bits can take one of two values: 0 or 1 to indicate failure or survival, respectively. A consecutive sequence of 0's is called a "0 run," and a consecutive sequence of 1's is called a "1 run." If the next bit in the sequence is a different value from the previous bit, e.g., a 1 follows a sequence of 0's, then that indicates either the beginning of a new run or the beginning of a phrase. If the bit subsequent to that last bit is the same value, i.e., a 1 follows the previous 1, then that indicates the beginning of a 1 run. If the subsequent bit is a different value, i.e., a 0 follows the previous 1, then that indicates the beginning of a phrase. The phrase is now comprised of at least two elements, the first value, i.e., the 1, and the differing second value, i.e., the 0. As the sequence continues with bits of the second value, the length of the phrase increases, until a bit of the first value appears, which then indicates the end of the phrase. Therefore, each phrase is constructed of two values: the first element of the phrase is one of the values and it is followed by a sequence of the other value. The dic-

tionary for the proposed algorithm is comprised of these phrases. Note that this definition of a phrase is different from the standard Lempel-Ziv definition.

Figure 3.1 shows the flowchart for the developed compression algorithm for a binary system with n components and with $\{MCS\}$ denoting the set of minimum cut sets (MCSs) of the system. The output of the compression algorithm is the compressed system CPT, $cCPT_{sys}$, and the accompanying dictionary of phrases, d_0 . Steps of the algorithm are described below.

For each row $k = 1, \dots, 2^n$ of the system CPT, the component states $s_1, \dots, s_n$ are computed in terms of the row number based on the specific pattern used in defining the table. The table is constructed with $C_1, \dots, C_n$ organized from left to right. Each row of the CPT is one of the mutually exclusive combinations of component states. The specific pattern we use to organize these states is: C_1 is in state 0 for rows $k = 1, \dots, 2^{n-1}$ and in state 1 for rows $k = 2^{n-1} + 1, \dots, 2^n$; C_2 is in state 0 for rows $k = 1, \dots, 2^{n-2}$, in state 1 for rows $k = 2^{n-2} + 1, \dots, 2^{n-1}$, in state 0 for rows $2^{n-1} + 1, \dots, 2^{n-1} + 2^{n-2}$, and in state 1 for rows $2^{n-1} + 2^{n-2} + 1, \dots, 2^n$; etc. This pattern continues through C_n , which is therefore in state 0 and 1 in alternating rows, i.e., in state 0 in odd rows and state 1 in even rows. Utilizing this pattern in constructing the CPT, we determine the state of component i , $i = 1, \dots, n$, in row k of the CPT according to the rule

$$s_i = \begin{cases} 0 & \text{if } \text{ceil}\left(\frac{k}{2^{n-i}}\right) \in \text{odd} \\ 1 & \text{if } \text{ceil}\left(\frac{k}{2^{n-i}}\right) \in \text{even} \end{cases} \quad (3.1)$$

where $\text{ceil}(x)$ is the value of x rounded up to the nearest integer. For example, for a system comprised of 20 components, the state of C_{15} in row 450,000 is 0 because $\text{ceil}\left(\frac{450000}{2^{20-15}}\right) = 14063$ is odd. Note that using the above rule, the data on component states in the CPT is easily removed without any loss of information.

For each row, the component states are checked against $\{MCS\}$ to determine the system state. If all of the components comprising at least one MCS have failed, then the system is in the fail state; if not all components in any of the MCSs have failed, then system is in the survival state. This value of the system state is then encoded in compressed form as a run or a phrase.

Runs can be 0 runs or 1 runs, and as the number of consecutive repeated values increases, the length of the run is increased. When a phrase is encountered, the phrase is checked against the contents of the dictionary to determine if the phrase is an existing one that has already been stored in the dictionary, or if it is a new one that must be added to the dictionary. Each phrase in the dictionary is fully defined by four variables: 1) the phrase number, p , 2) the first value in the phrase, v_1 , 3) the second and subsequent values in the phrase, v_2 , and 4) the length of the phrase L_p . Once the existing or new phrase has been

identified, the number of repeated instances of the particular phrase, denoted n_p , is updated.

Each row of the compressed CPT is comprised of three values: 1) an indicator variable that defines whether the row is the beginning of a run or a phrase; 2) if a run, the value r of the run; if a phrase, the phrase number p in the dictionary; and 3) if a run, the length L_r of the run; if a phrase, the number n_p of repeated instances of the phrase. Therefore, a run is defined by the values $\{\text{run}, r, L_r\}$ and a set of phrases is defined by the values $\{\text{phrase}, p, n_p\}$. Once all rows of the system CPT have been processed, the end result of the compression algorithm is the compressed CPT, $cCPT_{sys}$, and the dictionary, d_0 . The size of this data is typically orders of magnitude smaller than the size of the original CPT.

Algorithm A: Compression Algorithm

Input: $n, \{MCS\}$
 Output: $cCPT_{sys}, d_0$

for $k \leftarrow 1$ to 2^n , do

Figure 3.1: Flowchart of compression algorithm

3.4 Application to an example system

To illustrate the developed compression algorithm, we apply it to the example system shown in Figure 3.2, which is adopted from Bensi et al. (2013). The system consists of a parallel subsystem (C_1, C_2, C_3) and series subsystems (C_4, C_5, C_6) and (C_7, C_8) .

Figure 3.2: Example system

For this system, the set of MCSs is $\{MCS\} = \{(C_1, C_2, C_3, C_4), (C_1, C_2, C_3, C_5), (C_1, C_2, C_3, C_6), (C_7), (C_8)\}$. The BN formulation of the system is shown in Figure 3.3.

Figure 3.3: BN formulation of the example system

The full system CPT is shown in Table 3.1. The first 8 columns give the states of components $C_1, \dots, C_8$, constructed according to the pattern described in Section 3. The right-most column gives the state of the system given the states of the components in that row. Because the system is comprised of 8 components, the system CPT consists of $2^8 = 256$ rows.

Table 3.1: System CPT for example system

C_1	...	C_7	C_8	sys
0	...	0	0	0
0	...	0	1	0
0	...	1	0	0
0	...	1	1	0
$\vdots$		$\vdots$	$\vdots$	$\vdots$
1	...	0	0	0
1	...	0	1	0

1	...	1	0	0
1	...	1	1	1

3.4.1 Construction of compressed CPT

We now proceed with implementing the proposed compression algorithm as described in Section 3 with the flowchart shown in Figure 3.1 to compress the system CPT given in Table 3.1. Note that we only need to compress the vector of system states, i.e., the right-most column in the CPT. There is no need to compress the entire CPT as the component states in any row, i.e., the values in columns 1, ..., n , can be determined according to the rule given in Equation (3.1).

In row $k = 1$, $\{s_1, \dots, s_8\} = \{0, \dots, 0\}$. As all components in the system are in the failed state, clearly the system is also in the failed state. More rigorously, checking against $\{MCS\}$, we see that all components in the first MCS, (C_1, C_2, C_3, C_4) , have failed. Therefore, $sys = 0$. Note that once we find that all components in a MCS are in the failed state, we need not check the component states against the remaining MCSs as the failure of any one MCS indicates failure of the system.

As we continue through the rows $k = 2, \dots, 31$, we find the system to be in the failed state, until we reach row $k = 32$. Therefore, the compressed CPT begins with a 0 run of length 31. At $k = 32$, $\{s_1, \dots, s_8\} = \{0, 0, 0, 1, 1, 1, 1, 1\}$. Checking against $\{MCS\}$, we see that none of the MCSs have failed. In fact, the path from source to sink flows through C_4, C_5, C_6, C_7, C_8 . Thus, $sys = 1$, and we have reached the end of the initial 0 run. Looking at $k = 33$, $\{s_1, \dots, s_8\} = \{0, 0, 1, 0, 0, 0, 0, 0\}$. We see that the component in the fourth MCS (C_7) has failed. Therefore, $sys = 0$ and we have a phrase beginning at row $k = 32$. Had the value in row $k = 33$ been $sys = 1$, we would have had a 1 run.

At this point, we have not yet encountered any phrases and our dictionary is empty. Therefore, this is the beginning of a new phrase. For $k = 34$ and $k = 35$, $sys = 0$. At $k = 36$, $sys = 1$, indicating the end of the phrase. The full phrase, therefore, is $\{1, 0, 0, 0\}$ for $k = 32, \dots, 35$, which is stored in the dictionary as $\{1, 1, 0, 4\}$. These four values indicate that the phrase number is $p = 1$, the first value in the phrase is $v_1 = 1$, the second and subsequent values in the phrase are $v_2 = 0$, and the length of the phrase is $L_p = 4$. Note that in the binary case, v_1 and v_2 are complements, so knowing one value enables us to know the other. However, while it is not necessary to store both values in the initial construction of the compressed system CPT, in the subsequent process for inference, v_1 and v_2 can take values different from 0 and 1 and are no longer guaranteed to be complements. Therefore, during inference, we must store both v_1 and v_2 , and all four values to define the phrase are recorded in this stage as well for consistency.

After determining $sys = 1$ for $k = 36$, we continue for $k = 37, \dots, 39$ with the system state $sys = 0$ and for $k = 40$ with the system state $sys = 1$, thus we again encounter the phrase $sys = \{1, 0, 0, 0\}$. This is now an existing phrase that we call from the dictionary. Therefore, in the compressed system CPT, we reference phrase $p = 1$ and increase the number of instances of this phrase by 1. We continue through the remaining rows

$k = 40, \dots, 2^n = 256$. Once we have processed all the rows of the system CPT, the end result for the compressed CPT and the dictionary are as given in Tables 3.2 and 3.3, respectively.

Table 3.2: Compressed system CPT for example system

run or phrase	r or p	L_r or n_p
run	0	31
phrase	1	56
run	1	1

Table 3.3: Dictionary for example system

p	v_1	v_2	L_p
1	1	0	4

Tables 3.2 and 3.3 indicate that the total number of elements to be stored for the compressed system CPT is 9 and for the dictionary is 4 for a total of 13 elements. This is not counting the table headings, which are included for reference here and need not be stored when implementing the compression algorithm in a computer. Note that we can check that the total number of rows represented in the compressed CPT corresponds with the total number of rows of the system full CPT. From Tables 3.2 and 3.3 for the example system, we start with a run of length 31, have 56 instances of a phrase of length 4, and end with a run of length 1. This equals $31 + 56 \times 4 + 1 = 256$ rows of the original CPT that are now stored in the compressed CPT. Thus, all the information in the original CPT is stored in the compressed CPT and the compression is lossless. And the total number of elements that must be stored during the construction of the BN has been reduced from $2^8 = 256$ total elements in the original CPT to 13 in the compressed CPT using the developed compression algorithm. In a system with a large number of components, the amount of data reduction can be even more dramatic than that observed for this small example system.

4. Inference algorithm

4.1 Introduction

Once the Bayesian network (BN) model of a system has been constructed, inference is required to draw conclusions about the system. Statistical inference is the process of making conclusions about uncertain quantities in the system based on data or observations. In Bayesian inference, the notion of information updating is used. The estimated probability distribution of an uncertain variable or event is updated as new information becomes available.

The power of the BN is in its facilitation of both forward and backward inference. Forward inference is the calculation of the probability distribution of a node in the network based on assumed prior marginal and conditional distributions on all other nodes in the BN. This is also known as predictive inference in the sense that, given a particular prior state of the network, the resulting probability distribution over any node in the BN can be computed. An example of forward inference for system reliability analysis is the calculation of the overall system failure probability based on defined marginal and conditional failure probabilities for the components comprising the system.

Backward inference is the calculation of the posterior probability distribution of any node in the network given observations on one or more nodes in the BN. This is also known as diagnostic inference in the sense that once an event has been observed, the posterior probabilities of events that may have contributed to that observation can be computed. For example, in system reliability analysis, if failure of the system has been observed, the posterior probabilities of failure of individual components that may have led to the system failure can be computed using backward inference. While many techniques are available to perform forward inference, BNs are particularly powerful in their ability to perform backward inference and updating of system assessment in light of new information.

This chapter first describes two major inference algorithms, variable elimination and junction tree, then describes the proposed algorithm for inference. The algorithm is then applied to an example system to illustrate the details of the inference process and the effectiveness of the proposed method.

4.2 Existing inference algorithms

4.2.1 Background

There are both exact and approximate methods for inference. Approximate methods are generally sampling based, including importance sampling (Salmeron 2000, Yuan and Druzdzel 2006) and Markov chain Monte Carlo (Gilks et al. 1996, Beck and Au 2002) methods. In theory, these methods converge to the exact solution for a sufficiently large number of samples. In practice, however, the rate of convergence is unknown and can be slow (Straub 2009). This is especially true when simulating events that are a priori unlikely. In analyzing a reliable system where failure events are rare, the number of simulations required to achieve a sufficiently large sample for convergence can become prohibitive.

In addition, many engineering applications of BN methods for system reliability analysis require near-real time updating, e.g., for post-hazard system assessment and decision support. For these applications, approximate methods that are computationally intensive are not appropriate. Therefore, exact methods of inference are preferred and are considered here. In the following sections, the two major algorithms used for exact inference, the variable elimination (VE) algorithm and the junction tree (JT) algorithm, are described.

4.2.2 Variable elimination

In the VE algorithm (Dechter 1999), our objective is to perform inference on the “query” node, the node for which the updated probability distribution is of interest. To do this, we eliminate all the other nodes in the network, one by one, until we are left with the query node – hence the name of the algorithm, variable elimination. Elimination of each node corresponds to summing of the joint distribution over all states of the node to be eliminated, resulting in an intermediate factor λ that is used during the next step of elimination. For example, for the system shown in Figure 2.2 with components C_1 to C_n , suppose we are interested in the updated distribution of the state of component C_1 , given a particular state sys of the system. The VE calculation for this query is

$$\begin{aligned} p(C_1, sys) &= \sum_{C_2} \cdots \sum_{C_n} p(C_1) p(C_2) \cdots p(C_{n-1}) p(C_n) CPT_{sys} \\ &= p(C_1) \sum_{C_2} p(C_2) \cdots \sum_{C_{n-1}} p(C_{n-1}) \sum_{C_n} p(C_n) CPT_{sys} \\ &= p(C_1) \sum_{C_2} p(C_2) \cdots \sum_{C_{n-1}} p(C_{n-1}) \lambda_n \\ &\quad \cdots \\ &= p(C_1) \lambda_2 \end{aligned} \tag{4.1}$$

where CPT_{sys} is the conditional probability table for the system node in the BN and λ_i is the intermediate factor, in the form of a table, created after the elimination of node C_i . The conditional probability of interest is then obtained by dividing the joint probability by $p(sys)$ such that $p(C_1|sys) = p(C_1, sys)/p(sys)$. In the VE calculation, nodes $C_2, \dots, C_n$ have been eliminated to arrive at the query node, C_1 .

The intermediate factors λ_i that are calculated at each step during elimination need not be stored for subsequent steps in the elimination process. Therefore, there are fewer demands on memory storage compared to an algorithm like the JT, which will be described in the following section. However, because the λ_i 's are discarded and not stored for future use, there is in some cases the need to repeat computations for λ_i , e.g., in the case of multiple queries or when considering different evidence scenarios. This results in increased computation time. It is noted that the order in which the variables in a network are eliminated results in different memory storage and computation time requirements for the VE algorithm. The selection of an optimal elimination ordering, however, is an NP-hard problem. Depending on the structure of the network, heuristics can be developed to improve efficiency in elimination ordering. This topic is addressed in Chapter 5.

4.2.3 Junction tree

For a general BN, the JT algorithm (Spiegelhalter et al. 1993) improves on the VE algorithm by breaking down the network into smaller structures, called "cliques," which are subsets of the nodes in the BN. These cliques comprise the junction tree. The conditional probability tables associated with each clique are called "potentials," and the JT is initialized by computing the potentials over all cliques. These potentials are stored and reused in cases of updating, making the JT algorithm more efficient in computation time compared to an algorithm like the VE. However, large memory storage is required in the calculation of these clique potentials, and the size of a clique potential exponentially increases with the size of the clique. The advantage of the JT algorithm comes from breaking down the full BN into cliques. For a network structured as shown in Figure 2.2 with components C_1 to C_n , however, the JT is comprised of only one clique of size $n + 1$. As the size of the system increases, computation of the potential over this clique during the initialization of the JT increases exponentially and becomes intractable. With only one clique, there are no gains from using the JT compared to the VE. Of course there are alternative ways of formulating the BN model of the system that are more favorable to the JT algorithm. Bensi et al. (2013) have discussed these formulations and have proposed a BN topology optimization method to reduce the clique sizes in the BN. Nevertheless, even with these formulations, the JT algorithm becomes infeasible with increasing system size. Therefore, in this study the VE algorithm together with compression of data is used for inference.

4.3 Proposed inference algorithm

The proposed inference algorithm is developed based on the VE method. Variables are eliminated one by one to arrive at the query node, with the intermediate factor λ_i created

after the elimination of component C_i . The key to being able to perform inference on the BN constructed using the compression algorithm described in Chapter 3 is that the λ_i 's are also compressed using the same compression algorithm as for CPT_{sys} . Further, the inference algorithm is developed to handle both the compressed CPT_{sys} and λ_i without decompressing or recompressing.

Let CPT_{sys} = the conventional uncompressed full system CPT, and $cCPT_{sys}$ = the compressed system CPT constructed by running the compression algorithm. Similarly, λ_i = the uncompressed intermediate factor after elimination of the i th component, with $i = 0, n, \dots, 1$ being the order of elimination, and $c\lambda_i$ = the corresponding compressed intermediate factor. $i = 0$ indicates that no components have been eliminated. Therefore, $\lambda_0 = CPT_{sys}$ and $c\lambda_0 = cCPT_{sys}$. Let m_i = the number of rows in $c\lambda_i$.

During the variable elimination process, $c\lambda_i$ is constructed row by row without constructing the uncompressed λ_i . Similar to the rows of the compressed system CPT, each row j of $c\lambda_i$, $j = 1, \dots, m_i$, is comprised of three values: 1) an indicator variable that defines whether the row is the beginning of a run or a phrase; 2) if a run, the value r_i^j of the run; if a phrase, the phrase number p_i^j in the dictionary; and 3) if a run, the length $L_{r_i}^j$ of the run; if a phrase, the number $n_{p_i}^j$ of repeated instances of the phrase. Thus, if row j of $c\lambda_i$ defines the beginning of a run, $c\lambda_i^j = \{\text{run}, r_i^j, L_{r_i}^j\}$. If row j of $c\lambda_i$ defines the beginning of a phrase, $c\lambda_i^j = \{\text{phrase}, p_i^j, n_{p_i}^j\}$.

During the compression of CPT_{sys} , an accompanying dictionary d_0 is constructed, which defines the phrases present in $cCPT_{sys}$. Remember that for the binary case under consideration, each phrase is constructed of two values. The first element of the phrase is one of the values and it is followed by a sequence of the other value. This is because if the first value repeats, then the sequence would be considered a run. The dictionary is updated as new phrases are formed during the process of constructing $c\lambda_i$. Let d_i = the updated dictionary after construction of $c\lambda_i$. d_i is constructed row by row. Each row of d_i defines a phrase, which is fully defined by four variables: 1) the phrase number p , 2) the first value in the phrase, v_1 , 3) the second and subsequent values in the phrase, v_2 , and 4) the length of the phrase L_p .

For the developed inference algorithm, we begin with $cCPT_{sys}$ and d_0 . We then eliminate the components in the order $n, \dots, 1$. At each step, we construct $c\lambda_i$ from $c\lambda_{i+1}$ ($c\lambda_n$ from $c\lambda_0 = cCPT_{sys}$) and d_i from d_{i+1} (d_n from d_0) based on the algorithm described below. The construction of $c\lambda_i$ is performed row wise. At a high level, from $c\lambda_{i+1}$, we determine if we have a run or phrase and then compute the corresponding values of the run or phrase in $c\lambda_i$ accordingly. The dictionary is also updated, as necessary. The details of how this computation is performed are as follows.

Let $\{p_f\}$ = vector of length n defining the failure probabilities of components $C_1, \dots, C_n$ so that $p_{f,i} = \Pr(C_i = 0)$ is its i^{th} element. If the component nodes have parents, the par-

ent nodes are eliminated first so that the component failure probabilities are marginalized. We define two quantities: S_i^j = the run or phrase start row number in λ_i defined by row j of $c\lambda_i$, and R^j = remainder after processing row j of $c\lambda_i$ (defined in Table 4.1). Because the remainder is reset after the elimination of node i , it does not carry subscript i . Also, let Q define the set of query nodes (or components) and E denote the set of nodes (components) for which evidence is entered. The algorithm for constructing $c\lambda_i$ then is as follows:

Algorithm B: Inference Algorithm

Input: $n, \{p_f\}, cCPT_{sys}, d_0, Q, E$
Output: $\Pr(Q|E)$

For $i \leftarrow n$ down to 1, do

 If $i \in Q$

 Reorder λ_{i+1} such that component i is ordered to extreme left, i.e., numbered 1.
 $c\lambda_{i+1}^{\text{reordered}} \leftarrow \lambda_{i+1}^{\text{reordered}}$ using compression algorithm.
Do not increment i .

 Else for $j \leftarrow 1$ to m_{i+1} , do

 Switch {run, phrase}, $S_{i+1}^j \in \{\text{odd}, \text{even}\}$, $(L_{r_{i+1}}^j \text{ or } L_{p_{i+1}}^j) \in \{\text{odd}, \text{even}\}$.
Construct $c\lambda_i$ and d_i according to the case rules in Tables 4.1 and 4.2.
Increment i .

end

Tables 4.1 and 4.2 show the switch cases and the elements for constructing $c\lambda_i$ and d_i , respectively. In both tables, the switch cases are shown in the left three columns. In Table 4.1 for $c\lambda_i$, the fourth and fifth columns list the run or phrase values, and the last column is an update of the remainder for row j . The case for a run with $S_{i+1}^j \in \text{even}$ is special. It involves the remainder from the preceding row of $c\lambda_i$, as described below. The last column of the table lists the remainder value for the j^{th} row of $c\lambda_i$. This is updated for possible use for the next row of $c\lambda_i$ or d_i . In Table 4.2 for d_i , the last three columns list the first and second values of the new phrase and its length. When $S_{i+1}^j \in \text{even}$, the first phrase value involves the remainder corresponding to the preceding row of $c\lambda_i$.

Note that the length of a run, $L_{r_i}^j$, is guaranteed to be an integer due to the switch case based on the length of the run from the previous intermediate factor. I.e., if the previous run length is odd, the new run length is computed by subtracting 1, resulting in an even number, which is then divided by 2, producing an integer; while an even previous run length is simply divided by 2, producing an integer as well. Similarly, the length of a

phrase, $L_{p_i}^j$, is guaranteed to be an integer due to the switch case based on the length of the phrase from the previous intermediate factor, with odd phrase lengths subtracted by 3 or 1 before dividing by 2, and even phrase lengths subtracted by 2 before the division.

Finally, as mentioned above, in constructing $c\lambda_i$, the case of a run with $S_{i+1}^j \in \text{even}$ is special. When a run starts on an even row number in λ_i , the first value of the run is multiplied by $\Pr(C_i = 1)$, while subsequent pairs of values in the run are multiplied by $\Pr(C_i = 0)$ and $\Pr(C_i = 1)$, respectively. This first value must be taken into account as an additional run of unit length. Thus, for this special case, we concurrently construct rows j and $j + 1$ of $c\lambda_i$. For the first run, $r_i^j = r_{i+1}^j \times \Pr(C_i = 1) + R^{j-1}$, where R^{j-1} is the remainder from row $j - 1$ of $c\lambda_i$ and $L_{r_i}^j = 1$. For the second run, $r_i^{j+1} = r_{i+1}^j$ and $L_{r_i}^{j+1} = (L_{i+1}^j - 1)/2$. Because one row of $c\lambda_{i+1}$ now corresponds to two rows in $c\lambda_i$, subsequent row numbers in $c\lambda_i$ are incremented by a 1, or if there are multiple instances x of this special case, the row numbers are incremented by x .

With the rules described in Tables 4.1 and 4.2, the developed inference algorithm is thus able to handle both CPT_{sys} and λ_i in compressed form without decompressing or recompressing. This enables the savings in memory storage requirement achieved from implementing the compression algorithm during the construction of the BN to be preserved through the process of inference.

Table 4.1: Rules for constructing $c\lambda_i^j$

switch			r_i^j or p_i^j	$L_{r_i}^j$ or $n_{p_i}^j$	R^j
run	$S_{i+1}^j \in \text{odd}$	$L_{r_{i+1}}^j \in \text{odd}$	r_{i+1}^j	$(L_{i+1}^j - 1)/2$	$r_{i+1}^j \times \Pr(C_i = 0)$
		$L_{r_{i+1}}^j \in \text{even}$	r_{i+1}^j	$L_{i+1}^j/2$	0
	$S_{i+1}^j \in \text{even}$	$L_{r_{i+1}}^j \in \text{odd}$	$r_{i+1}^j \times \Pr(C_i = 1) + R^{j-1}$ (also $r_i^{j+1} = r_{i+1}^j$)	1 (also $L_{r_i}^{j+1} = (L_{i+1}^j - 1)/2$)	0
		$L_{r_{i+1}}^j \in \text{even}$	$r_{i+1}^j \times \Pr(C_i = 1) + R^{j-1}$ (also $r_i^{j+1} = r_{i+1}^j$)	1 (also $L_{r_i}^{j+1} = (L_{i+1}^j - 2)/2$)	$r_{i+1}^j \times \Pr(C_i = 0)$
phrase	$S_{i+1}^j \in \text{odd}$	$L_{p_{i+1}}^j \in \text{odd}$	p_{i+1}^j	$n_{p_{i+1}}^j$	$v_{2_{i+1}}^j \times \Pr(C_i = 0)$
		$L_{p_{i+1}}^j \in \text{even}$	p_{i+1}^j	$n_{p_{i+1}}^j$	0
	$S_{i+1}^j \in \text{even}$	$L_{p_{i+1}}^j \in \text{odd}$	p_{i+1}^j	$n_{p_{i+1}}^j$	0
		$L_{p_{i+1}}^j \in \text{even}$	p_{i+1}^j	$n_{p_{i+1}}^j$	$v_{2_{i+1}}^j \times \Pr(C_i = 0)$

Table 4.2: Updating d_i for a new phrase starting in row j of $c\lambda_i$

switch			$v_{1_i}^j$	$v_{2_i}^j$	$L_{p_i}^j$
phrase	$S_{i+1}^j \in \text{odd}$	$L_{p_{i+1}}^j \in \text{odd}$	$\lceil v_{1_{i+1}}^j \times \Pr(C_i = 0) \rceil + \lceil v_{2_{i+1}}^j \times \Pr(C_i = 1) \rceil$	$v_{2_{i+1}}^j$	$\lceil (L_{i+1}^j - 3)/2 \rceil + 1$
		$L_{p_{i+1}}^j \in \text{even}$	$\lceil v_{1_{i+1}}^j \times \Pr(C_i = 0) \rceil + \lceil v_{2_{i+1}}^j \times \Pr(C_i = 1) \rceil$	$v_{2_{i+1}}^j$	$\lceil (L_{i+1}^j - 2)/2 \rceil + 1$
	$S_{i+1}^j \in \text{even}$	$L_{p_{i+1}}^j \in \text{odd}$	$R^{j-1} + \lceil v_{1_{i+1}}^j \times \Pr(C_i = 1) \rceil$	$v_{2_{i+1}}^j$	$\lceil (L_{i+1}^j - 1)/2 \rceil + 1$
		$L_{p_{i+1}}^j \in \text{even}$	$R^{j-1} + \lceil v_{1_{i+1}}^j \times \Pr(C_i = 1) \rceil$	$v_{2_{i+1}}^j$	$\lceil (L_{i+1}^j - 2)/2 \rceil + 1$

4.4 Example system

To illustrate the proposed inference algorithm, we apply it to the example system shown in Figure 3.2. We define the prior probabilities of failure as 0.2 for the parallel components and 0.01 for the series components, such that $\{p_f\} = \{0.2, 0.2, 0.2, 0.01, 0.01, 0.01, 0.01, 0.01\}$ for components $C_1, \dots, C_8$.

4.4.1 Results of implementing inference algorithm

We begin with the compressed system CPT $cCPT_{sys}$ given in Table 3.2 and the accompanying initial dictionary d_0 given in Table 3.3. Suppose we are interested in the backward inference problem of obtaining the posterior probability distribution of C_1 given the observation that the system has failed. Our elimination order is 8, ..., 1.

In the first step of the elimination process, we are eliminating C_8 . For reference, the uncompressed intermediate factor λ_8 created after this elimination is shown in Table 4.3.

Table 4.3: Uncompressed intermediate factor λ_8 constructed after elimination of C_8 in the example system

C_1	...	C_6	C_7	λ_8
0	...	0	0	0
0	...	0	1	0
0	...	1	0	0
0	...	1	1	0
$\vdots$		$\vdots$	$\vdots$	$\vdots$
1	...	0	0	0
1	...	0	1	0.99
1	...	1	0	0
1	...	1	1	0.99

For the proposed inference algorithm, however, we need not work with matrices in uncompressed form. We will be constructing $c\lambda_8$ directly from $cCPT_{sys}$ as follows. $cCPT_{sys}$ consists of 3 rows; therefore, $m_{sys} = 3$. Looking at row $j = 1$ of $cCPT_{sys}$, we see that it is a run; we are starting in row 1, so $S_{sys}^1 \in \text{odd}$; and the length of the run is 31, so $L_{r_{sys}}^1 \in \text{odd}$. Therefore, in constructing $c\lambda_8^1$, we use the rules in the first row of Table 4.1: $r_8^1 = r_{sys}^1 = 0$, $L_{r_8}^1 = (L_{sys}^1 - 1)/2 = (31 - 1)/2 = 15$, and we have a remainder $R^1 = r_{sys}^1 \times \Pr(C_8 = 0) = 0$.

Moving to row $j = 2$, we have a phrase that starts in row 32 with a phrase length of 4. Therefore, we use the last row of Table 4.1 to construct $c\lambda_8^2$: $p_8^2 = p_{sys}^2 = 1$, $n_{p_8}^2 = n_{p_{sys}}^2 = 56$, and we have a remainder $R^2 = v_{2_{sys}}^2 \times \Pr(C_8 = 0) = 0$. Because we are now dealing with phrases, we also need to update our dictionary d_8 with this new phrase start-

ing in row $j = 2$ of $c\lambda_8$. Given the even starting row number and even phrase length, we are again using the last row, now in Table 4.2, to update d_8 : $v_{1_8}^2 = R^1 + \left[v_{1_{sys}}^2 \times \Pr(C_8 = 1) \right] = 0 + [1 \times 0.99] = 0.99$, $v_{2_8}^2 = v_{2_{sys}}^2 = 0$ and $L_{p_8}^2 = \left[(L_{sys}^2 - 2) / 2 \right] + 1 = 2$.

Here, we see that the run values r and the values of the phrases v_1 and v_2 can take on values other than 0 or 1, even in the binary case, as we multiply by component failure probabilities that are different from 0 or 1. However, the intermediate factors λ_i can still be compressed and the inference algorithm on the compressed matrices still works because the calculations for each row only involve the failure probability of one component so that the number of values that the runs and phrases can take on is still finite.

Finally, moving to row $j = 3$, we have a run that starts in row 256 with a run length of 1. Therefore, we use the third row of Table 4.1 to construct $c\lambda_8^3$: $r_8^3 = r_{sys}^3 \times \Pr(C_8 = 1) + R^2 = (1 \times 0.99) + 0 = 0.99$, and $L_{r_8}^3 = 1$. In this case, because the run is only of length 1, we only have a value for the first run, and there is no second run that needs to be taken into account for the special case of $S_{sys}^3 \in \text{even}$. The end result is the constructed $c\lambda_8$ and d_8 as shown in Tables 4.4 and 4.5, respectively.

Table 4.4: Compressed intermediate factor $c\lambda_8$ constructed after elimination of C_8 in the example system

run or phrase	r or p	L_r or n_p
run	0	15
phrase	1	56
run	0.99	1

Table 4.5: Dictionary d_8 constructed after elimination of C_8 in the example system

p	v_1	v_2	L_p
1	0.99	0	2

We continue the elimination process until we have eliminated the variables $C_2, \dots, C_8$ and arrive at the query node of interest C_1 . Our original inference question was to obtain the posterior probability distribution of C_1 given an observation of system failure. From the VE process, we obtain the joint probability $p(C_1, \text{sys})$, which we divide by $p(\text{sys})$ to arrive at the final result: $p(C_1 | \text{sys} = 0) = [0.2093 \ 0.7907]$ for C_1 being in the failure or survival state, respectively. Thus, given the observation that the system has failed, the probability of failure for C_1 has been updated from a prior failure probability of 0.2 to a posterior failure probability of 0.2093.

Comparing the uncompressed intermediate factor λ_8 given in Table 4.3 with the compressed $c\lambda_8$ and accompanying dictionary d_8 given in Tables 4.4 and 4.5, respectively, we see that the memory storage requirements have been reduced from $2^7 = 128$ elements to a total of 13 elements, 9 for $c\lambda_8$ and 4 for d_8 . As described in Section 3.4.1, we

are able to check that the total number of rows represented in $c\lambda_8$ corresponds with the total number of rows of the full λ_8 . In $c\lambda_8$, we have a run of length 15, have 56 instances of a phrase of length 2, and end with a run of length 1. This equals $15 + 56 \times 2 + 1 = 128$ rows in the uncompressed intermediate factor that are now stored in compressed form. Thus, the compression continues to be lossless, and we again emphasize that we do not need to decompress $cCPT_{sys}$ or any of the λ_i values to compute $c\lambda_i$ for the inference algorithm. Similar reductions in memory storage are achieved at each step of the elimination process for inference.

4.5 Dependent components

The inference algorithm presented in the previous section assumes independence between the components of the system. In this section, we discuss the treatment of dependent components, i.e., components with common parents representing, for example, common hazards or demands.

4.5.1 Algorithm for dependent components

In performing inference on a system with dependent components, we first eliminate the parent nodes. Let $P_1, \dots, P_{n_p}$ denote the parent nodes, where n_p indicates the total number of parents, of n components $C_1, \dots, C_n$. Let n_j denote the number of components that share the common parent P_j , so that $C_{1_j}, \dots, C_{n_j}$ is the subset of components that are the children of P_j . A parent node P_j is eliminated by summing over the states P_j according to the total probability rule

$$\lambda_{P_j} = \sum_{P_j} p(C_{1_j}|P_j) \cdots p(C_{n_j}|P_j) p(P_j) \quad (4.2)$$

where use has been made of the conditional independence of the component states. This elimination creates the intermediate factor λ_{P_j} . For VE, all nodes other than the query node, including parent nodes in the network, must be eliminated to arrive at the inference results. Therefore, for components with multiple parents, the parents are still eliminated one by one, according to Equation (4.2).

The key to preserving the memory storage savings achieved by the compression algorithm during the process for inference is that the intermediate factors created after the elimination of the parent nodes are also compressed using runs and phrases as in the compression algorithm described in Chapter 3. That is, λ_{P_j} is stored as $c\lambda_{P_j}$. Although the values stored in λ_{P_j} are no longer binary, i.e., no longer just 0 or 1, they are a finite set of values and can be compressed as $c\lambda_{P_j}$. Note that the values in λ_{P_j} are obtained by multiplying together the conditional distributions of the children components given the parent and the marginal distribution of the parent. Since each of these has finite values, the values appearing in λ_{P_j} are also finite. In particular, when the conditional failure

probabilities of the components are equal across the binary children of P_j , the maximum number of distinct values appearing in λ_{P_j} is $n_j + 1$. This is because for each combination of component states, the value in λ_{P_j} is based on the number of components that are in the failed state for that combination, with $n_j + 1$ possible numbers of component failures. For more general cases where the conditional failure probabilities across the children components vary, the number of possible distinct values appearing in λ_{P_j} increases. Although it is reasonable to assume that subsets of components have similar component failure probabilities, further study is needed to address the general case. One alternative is that the intermediate factor created after elimination of the parent node need not be compressed. This will work if there are not too many components that share the same parent node. Standard VE can be used to eliminate the parent nodes, and then the inference process proceeds as described below.

Once the parent nodes have been eliminated, the components must be eliminated. During the elimination of the component nodes, if the component being eliminated does not have any parents, the VE is performed as described in Section 4.3. If the component does have parents, then we must account for the intermediate factors created after the elimination of the parents of that component. Let n_i denote the number of parents of component C_i such that $P_{1_i}, \dots, P_{n_i}$ is the subset of nodes that are parents to C_i . Also let $\hat{C}$ denote the set of components with parents. When eliminating a component $C_i \in \hat{C}$, both the intermediate factor created from the previous elimination step, λ_{i+1} , and the intermediate factors created from the elimination of its parent nodes are taken into account. Thus, the elimination of C_i during the variable elimination process for inference is performed according to the equation

$$\lambda_i = \sum_{C_i} \lambda_{P_{1_i}} \cdots \lambda_{P_{n_i}} \lambda_{i+1} \quad (4.3)$$

This results in the intermediate factor λ_i , which is used in the subsequent elimination step. Note that λ_i is also stored in compressed form as $c\lambda_i$ using the developed compression algorithm during the inference process. As was the case for the inference algorithm presented in Section 4.3 and in the elimination of the parent nodes described above, λ_i , though no longer binary, is comprised of a limited number of values, and can therefore be compressed as $c\lambda_i$.

Let $\{p_f\}$ denote the prior probabilities of failure of components $C_1, \dots, C_n$, including the conditional failure probabilities of components with parents. Note that rather than a vector defining the failure probabilities, this is now a set of tables, as the component failure probabilities are defined for all possible combinations of parent states. To perform inference on query Q given evidence E for a system with dependent components, we employ the algorithm described below. The mentioned compression algorithm is Algorithm A: Compression Algorithm described in Section 3.3, and the VE inference algorithm refers to Algorithm B: Inference Algorithm described in Section 4.3.

Algorithm C: Algorithm for Dependent Components

Input: $n, \{p_f\}, cCPT_{sys}, d_0, Q, E$

Output: $\Pr(Q|E)$

For $j \leftarrow 1$ to n_P , do

 Eliminate P_j according to Equation (4.2).

$c\lambda_{P_j} \leftarrow \lambda_{P_j}$ using compression algorithm.

end

For $i \leftarrow n$ down to 1, do

 If $i \in \hat{\mathcal{C}}$

 Compute λ_i according to Equation (4.3).

$c\lambda_i \leftarrow \lambda_i$ using compression algorithm.

 Else

 Eliminate components using VE inference algorithm.

end

An example using this algorithm to perform inference on a system with dependent components that share a common hazard parent node is described in Chapter 5.

5. Performance of algorithms

5.1 Introduction

In this chapter, we apply the proposed compression and inference algorithms described in the previous chapters to two test example systems. We compare the performance of the new algorithms versus existing methods in terms of both memory storage and computation time. We then present several heuristic augmentations to improve the performance of the proposed algorithms. These include heuristics for the compression algorithm, for the inference algorithm, and a multi-scale modeling approach that employs supercomponents. The gains achieved by implementing these heuristics, particularly in computational efficiency, are demonstrated. Finally, the algorithms are applied to a power system application to illustrate the effectiveness of the proposed methodology for reliability assessment of a relatively large infrastructure system.

5.2 Test example systems

We begin with the example system adopted from Bensi et al. (2013) shown in Figure 3.2, with the Bayesian network (BN) formulation of the system as shown in Figure 3.3. The system is comprised of 8 components and consists of a parallel subsystem, $\{C_1, C_2, C_3\}$, and two series subsystems, $\{C_4, C_5, C_6\}$ and $\{C_7, C_8\}$. Because the objective is to see how the proposed algorithms scale, we increase the number of components in the first two subsystems to a total number of components in the system n . The systems obtained by increasing the number of components in the series and parallel subsystems are shown in Figures 5.1 and 5.2, respectively. The BN formulation of these expanded systems is shown in Figure 5.3.

Figure 5.1: Example test system: expanded with increased number of components in series subsystem

Figure 5.2: Example test system: expanded with increased number of components in parallel subsystem

Figure 5.3: BN formulation of expanded example systems

The resulting analyses of these two expanded systems demonstrates how the proposed algorithms perform compared to existing algorithms for systems of increasing size. We note that, as pointed out by Der Kiureghian and Song (2008), Song and Ok (2010) and Bensi et al. (2013), the above systems can be more efficiently represented as a system of three supercomponents, each supercomponent representing a series or parallel subsystem. However, here we disregard this advantage in order to investigate the system size effect. Supercomponents will be discussed in detail in Section 5.3.4.

5.2.1 Inference

To perform inference on the example system, we first compress the system conditional probability table (CPT) using the compression algorithm presented in Chapter 3. The process for constructing the compressed system CPT for the basic case of $n = 8$ is described in Section 3.4. For compressing the system CPTs for the expanded systems, the

minimum cut sets (MCSs) of the system shown in Figure 5.1 with an increased number of components in the series subsystem are $\{(C_1, C_2, C_3, C_6), \dots, (C_1, C_2, C_3, C_n), (C_4), (C_5)\}$, and the MCSs of the system shown in Figure 5.2 with an increased number of components in the parallel subsystem are $\{(C_1, \dots, C_{n-5}, C_{n-4}), (C_1, \dots, C_{n-5}, C_{n-3}), (C_1, \dots, C_{n-5}, C_{n-2}), (C_{n-1}), (C_n)\}$.

Following the example system given in Chapter 4 to illustrate the inference algorithm, the BN is initialized with a prior probability of failure for the parallel components, C_1, C_2, C_3 in Figure 5.1, and $C_1, \dots, C_{n-5}$ in Figure 5.2, of 0.2, and a prior probability of failure for the components in series, $C_4, \dots, C_n$ in Figure 5.1, and $C_{n-4}, \dots, C_n$ in Figure 5.2, of 0.01. We are interested in updating the probabilities of failure of the system and components given new information, or evidence. We first implement the compression algorithm to construct the BN, then utilize the inference algorithm to perform variable elimination (VE) on the compressed matrices.

For illustration purposes, in the following we conduct inference using component 1. The states 0 or 1 indicate failure or survival, respectively. Figure 5.4 shows the results for the forward inference problem, with the updated probabilities of system failure given the evidence that component 1 has failed, i.e., $\Pr(\text{sys} = 0 \mid C_1 = 0)$. Figure 5.5 shows the results for the backward inference problem, with the updated probabilities of failure of component 1 given the evidence that the system has failed, i.e., $\Pr(C_1 = 0 \mid \text{sys} = 0)$. The terms forward and backward inference, used throughout this chapter, are as described in Section 4.1. That is, for the system reliability assessment performed in this study, forward inference refers to the calculation of the system failure probability conditioned on known states of one or more components, while backward inference computes posterior probabilities of failure of individual components once the system state has been observed. In Figures 5.4 and 5.5, these updated probabilities are plotted as a function of increasing system size, as indicated by the increasing number of total components in the system, n .

Figure 5.4: Results of forward inference: updated probabilities of system failure given component failure as a function of increasing system size

Figure 5.5: Results of backward inference: updated probabilities of component failure given system failure as a function of increasing system size

“Increase series” indicates the results from increasing the number of components in the series subsystem (Figure 5.1), and “increase parallel” indicates the results from increasing the number of components in the parallel subsystem (Figure 5.2). In Figure 5.4, we see that the updated probability that the system fails increases as the number of components in the series subsystem increases, as there are more MCSs that can fail to lead to system failure. In contrast, as the number of components in the parallel subsystem increases, for a system of 11 components or more, the updated probability of system failure remains essentially constant. This is because the probability of failure of MCSs involving the increasing number of parallel components becomes small, and the system failure probability becomes dominated by the failure probabilities of the two single-component MCSs $\{C_{n-1}\}$ and $\{C_n\}$.

In Figure 5.5, we see that as the number of components in the series subsystem increases, the conditional probability that C_1 has failed given that the system has failed increases. With an increased number of components in the series subsystem, there are an increased number of MCSs that involve component 1, i.e., $\{C_1, C_2, C_3, C_i\}$, $i = 6, \dots, n$. Since failure of any of these MCSs leads to system failure, an increase in the system size gives a higher probability that failure of component C_1 was “necessary” for the system to fail. In contrast, as the number of components in the parallel subsystem increases, the probability of failure for component C_1 given system failure again converges for a system of 11 components or more. In this case, the evidence on the system state is not informative for the component, and the updated probability of failure converges to the prior probability of failure of component C_1 . This is because the parallel system being highly reliable, it is unlikely to have contributed to the system failure. Figures 5.4 and 5.5 show that the proposed algorithms successfully perform both forward and backward inference.

5.2.2 Dependent components

For illustration of the treatment of dependent components, we add parent nodes to a subset of the components of the system. Let us consider the system given in Figure 3.2. Suppose the components in the parallel subsystem $\{C_1, C_2, C_3\}$ are subject to a common hazard indicated by the parent node P . Figure 5.6 shows the BN representing this system.

Figure 5.6: BN formulation of example system with dependent components

The probability of failure for the parallel components is dependent upon the common hazard. The BN is initialized such that without the hazard occurring, the failure probability each component is as before, 0.2. However, if the hazard occurs, the failure probability of components C_1, C_2 and C_3 increases to 0.4. Now, suppose we have evidence that component 1 has failed. We use the algorithm for dependent components presented in Section 4.5.1 to perform inference for this system. Figure 5.7 shows the inference results varying the prior hazard probability from 0.05 to 0.25.

Figure 5.7: Inference results given failure of component 1 for example system with dependent components varying hazard probability

Figure 5.7 shows the effect of the varying hazard probability on the inference results. To perform the inference, the parent node is eliminated first, followed by the individual components through the VE process with compressed CPTs and intermediate factors. The evidence on component 1 propagates through the network, producing: an updated hazard probability, an updated probability of failure of the other dependent components C_2 and C_3 , and an updated system failure probability. The updated system failure probability is relatively insensitive to the hazard probability, since the latter only affects the components in the parallel subsystem. This analysis enables us to update our assessment of the system state given observations among dependent components, perform post-hazard risk assessment, as well as analyze the reliability of a system under varying hazard scenarios.

5.2.3 Memory storage

In analyzing the ability of the proposed new algorithms to analyze systems of increasing size, we examine the performance of the new algorithms compared to existing methods on two measures: memory storage and computation time. Looking at memory storage first, Figure 5.8 shows the maximum number of values that must be stored in memory during the running of the algorithms, which is used as a proxy to assess the memory storage requirements of each algorithm. The algorithms are run in Matlab v7.10 on a 32 Gb RAM computer.

Figure 5.8: Memory storage requirements for proposed new algorithm compared to existing method as a function of system size

In Figure 5.8, the values for the “New” proposed algorithm are the maximum from the initial construction of the BN (including elements in both the compressed system CPT and the accompanying phrase dictionary) and during the inference process (including elements in both the intermediate factors λ_i and the accompanying phrase dictionaries at each elimination step). The values for the “Existing” algorithm indicate the maximum number of elements stored using the existing junction tree (JT) algorithm, as implemented in the Bayes Net Toolbox by Murphy (2001), and the naïve formulation shown in

Figure 5.3. The “X” mark indicates the maximum size of the system after which the existing algorithm can no longer be used because the memory demand exceeds the available memory storage capacity. These results are the same for both the system obtained from increasing the number of components in the series subsystem (Figure 5.1), and from increasing the number of components in the parallel subsystem (Figure 5.2).

Figure 5.8 shows that the proposed new algorithm achieves significant gains in memory storage demand compared to the existing algorithm. For the existing JT algorithm, the memory storage demand, as measured by the maximum number of values that must be stored, increases exponentially with the number of components in the system. In fact, the algorithm runs out of memory for a system comprised of more than 24 components. The memory storage demand using the proposed new algorithm not only does not increase exponentially, but remains constant, even as the number of components in the system increases. The total number of values stored is 15, compared to 2^{n+1} for the size of the full system CPT.

5.2.4 Computation time

In examining the performance of an algorithm, one cannot only look at memory storage demand. An analysis of computational efficiency is also required. Figure 5.9 shows the computation times required to run the algorithms as a function of increasing system size.

Figure 5.9: Computation times for proposed new algorithm compared to existing method as a function of system size

In Figure 5.9, the computation times are broken into the various functions for each algorithm. The bars labeled “New - compression” indicate the time required to compress the system CPT using the proposed compression algorithm. The next two blue bars indicate

the time required to perform forward inference on the system given $\{C_1 = 0\}$ and the time to perform backward inference on the component C_1 given $\{sys = 0\}$, respectively, using the proposed compression algorithm based on the VE method. The bars labeled “Existing - initialization” indicate the times required to initialize the BN using the existing JT algorithm. The next two red bars indicate the times required to perform forward inference and backward inference using JT. The computation times are recorded for systems of increasing size, as indicated by the total number of components in the system. The results given in Figure 5.9 are for the case where the size of the system increases by increasing the number of components in the series subsystem.

Figure 5.10 compares the computation times for the proposed algorithm for the two systems in Figures 5.1 and 5.2. Again the computation times are broken down into the component for compressing the system CPT, to perform forward inference on the system given the state of component C_1 , and to perform backward inference on component C_1 given the state of the system. The solid bars are for the case where the size of the system is increased by increasing the number of components in the series subsystem, and the bars with diagonal hatching are for the case where the size of the system is increased by increasing the number of components in the parallel subsystem.

Figure 5.10: Computation times for proposed new algorithm when increasing the number of components in the series compared to parallel subsystems

In Figure 5.10, we see that the algorithm performs slightly better in the initial compression of the system CPT and in both inference scenarios for the systems with an increased number of parallel components compared to series components. The algorithm, therefore, is better suited to systems that can be formulated as few MCSs comprised of many components each, compared to systems formulated as many MCSs of few components each. In the latter case, it is preferable to use an MLS formulation of the system.

Looking at Figures 5.8 and 5.9 together to examine both memory storage and computational efficiency, we see the classic storage space-computation time trade-off as described in Dechter (1999). Figure 5.8 shows the significant gains in memory storage demand achieved by the proposed new algorithm compared to the existing JT algorithm; while in Figure 5.9, we see that the new algorithm requires longer computation times than the existing algorithm. We note, however, that as systems become larger, i.e., for systems comprised of more than 20 components, the time to perform inference for both the new and existing algorithms is exponentially increasing with the system size. For the existing algorithm, this is due to the increasing clique size for the JT. For the new algorithm, the increase in computation time is due to the computations needed for compressing the CPTs and performing inference with compressed intermediate factors λ_i during the variable elimination process.

It is important to note that the natures of the memory and time constraints are fundamentally different. Memory storage is a hard constraint. If the maximum size of a CPT exceeds the storage capacity of a program or machine, no analysis can be performed. While it is true that memory can be distributed, e.g., in cloud storage, there still exists a hard limit on the maximum. In contrast, computation time is more flexible. Indeed, various recourses are available to address the computational time, such as parallel computing.

Based on the above analysis, we can claim that some system reliability problems that simply could not be solved using existing algorithms because of limited computer memory now can be solved using the proposed algorithms. Even so, the accompanying increase in computation time can sometimes be prohibitive, particularly when considering the scale of real-world infrastructure systems. The following sections describe methods developed to improve the computational efficiency of the proposed algorithms such that the algorithms achieve gains along both memory storage and computation time measures.

5.3 Heuristic augmentations

In this section, three heuristic augmentations to the proposed algorithms are presented. The first two utilize the ordering of the components for both the compression and inference algorithms, while the third addresses the BN formulation of the system to improve computational efficiency.

5.3.1 Heuristic for compression algorithm

To compress the full system CPT, the compression algorithm must run through each of the 2^n distinct combinations of component states (rows of the CPT). This leads to an exponentially increasing computation time for compressing the system CPT. However, knowledge about the structure of the system can be used to reduce the number of rows to be analyzed. For example, if component C_i on its own constitutes a MCS, i.e., failure of

C_i leads to system failure, we need not check the states of other components when C_i is in the failed state.

In general, as discussed in Chapter 4, determining the optimal ordering of nodes in a BN is an NP-hard problem. The heuristic employed here is to order the MCSs by size, and order the components such that those in the smallest MCSs are numbered first and appear to the left in the CPT. Knowing where these components appear in the CPT enables us to know which rows in the CPT need not be processed when running the compression algorithm. In general, given the states of the components $s_1, \dots, s_n$ for a system of n components, the corresponding row number r of the CPT is determined according to the rule

$$r = 1 + \sum_{i=1}^n s_i \times 2^{n-i} \quad (5.1)$$

Thus, given a particular ordering of components comprising an MCS of a known size, we can easily determine the rows where the system is in the failed state. For example, if component C_1 constitutes an MCS on its own, we know that for rows $1, \dots, 2^{n-1}$ the system is in the failed state. If, instead, components C_1 and C_2 constitute a MCS, then we know that for rows $1, \dots, 2^{n-2}$ the system is in the failed state. We call these “0 intervals,” i.e., intervals of rows in the system column of the CPT that are in the 0 state.

Figure 5.11 shows the flowchart of the proposed compression algorithm with this heuristic implemented. In the first step, if the row number k is determined to be within the 0 interval, then we need not continue through the bulk of the algorithm process. That is, we need not compute the states s_i , check against the set of MCSs $\{MCS\}$ to compute the system state, determine if the state is part of a run or a phrase, etc. Instead, we know that in the 0 interval, we are in a 0 run and we are able to directly move to updating the length of the 0 run in the CPT. Therefore, with this heuristic implemented, the full compression algorithm need only process through the rows between the 0 intervals. This reduces the number of rows that are processed and results in reduced time spent on compression.

Algorithm D: Compression Algorithm II

Input: $n, \{MCS\}$
Output: $cCPT_{sys}, d_0$

for $k \leftarrow 1$ to 2^n , do

Figure 5.11: Compression algorithm flowchart with component-order heuristic implemented

Figure 5.12 shows the result of applying this heuristic to the example systems presented in Section 5.2. The dashed and solid bars show the computation times required to compress the system CPT without and with the heuristic, respectively. In the example systems, the two components comprising the second series subsystem each on their own comprise a MCS. Therefore, these components are numbered first and appear to the left in the CPT. Specifically, for the system in Figure 5.1, components C_4 and C_5 are renumbered 1 and 2, and for the system in Figure 5.2, components C_{n-1} and C_n are renumbered components 1 and 2.

Figure 5.12: Computation times to compress the system CPT without versus with heuristic implemented

In Figure 5.12, we see the reduction in computation times achieved by employing the heuristic for the compression algorithm. In addition, comparing the results for systems with increasing number of components in the series versus parallel subsystems, we see that, consistent with the results in Figure 5.10, the algorithm performs better for the latter case. Thus, the algorithm is better suited to systems formulated as few MCSs of many components each, compared to systems formulated as many MCSs of few components each.

5.3.2 Heuristic for inference algorithm

During the VE process for inference, the selection of an optimal elimination ordering is an NP-hard problem. Similar to the heuristic for the compression algorithm, the heuristic for the inference algorithm looks to ordering the components in a particular way to improve computational efficiency without solving the optimization problem.

In VE inference, all nodes other than the query node must be eliminated to arrive at the posterior probability distribution of the query node. As described in Chapter 4, during the node elimination process, when we arrive at the query node, it is necessary to move it to the very left end of the CPT. This requires reordering of the elements in the intermediate factor λ_i , which is a computationally demanding effort. The heuristic employed here is to

order the components such that query components appear as far to the left in the CPT as possible. This minimizes the number of operations that must be performed to reorder λ_i . During reordering, the inference algorithm must process through each row of λ_i . For each component that is eliminated, the size of λ_i , which is represented in compressed form, decreases by a factor of 2. Thus, if query components are to the left in the CPT and are reached late in the elimination order, the number of rows to process to reorder λ_i is reduced.

Figure 5.13 shows the result of applying this heuristic to the example systems presented in Section 5.2. The computation times for forward and backward inference in systems with an increasing number of components in the series and parallel subsystems are plotted. Figures 5.13(a), 5.13(b), and 5.13(c) respectively show the results from using the existing JT algorithm and the proposed new algorithm without and with the heuristic employed. The C_1 labeled in the figures on which inference is performed refers to the component numbered C_1 in Figures 5.1 and 5.2. When implementing the heuristics for running the proposed algorithms, the compression algorithm heuristic results in the two components in the second series subsystem being renumbered as C_1 and C_2 . Therefore, in implementing the inference algorithm heuristic, which numbers query components to appear as far to the left in the CPT as possible, the query node is ordered next as C_3 in the reordered formulation of the system.

Figure 5.13: Computation times for forward and backward inference using the existing algorithm (a) and the new inference algorithms without (b) and with (c) the heuristic implemented

Comparing Figures 5.13(a) and 5.13(b), we see that the new algorithm without the heuristic employed requires longer computation times for inference than the existing JT algorithm. In addition, the computation times for both algorithms increase exponentially as the system size increases. However, in Figure 6(c) we see that with the heuristic employed, the new algorithm achieves computation times that are orders of magnitude smaller than either of the other algorithms: four orders of magnitude faster than the new algorithm without the heuristic employed, and three orders of magnitude faster than the existing JT algorithm.

In addition, and more importantly given the effect for large systems, the computation times are linearly, not exponentially, increasing with system size. The reason for this is

that when the heuristic is employed, the computation time becomes a function of not the full size of the λ_i 's, which exponentially increase with the system size, but a function of the size of the compressed λ_i 's, which we have seen remain constant with increasing system size. With the memory storage savings already demonstrated, these heuristics utilizing a more effective ordering of the components significantly improve the computational efficiency of both the compression and inference algorithms.

5.3.3 Supercomponents

In the previous sections, analyses of the example systems from Section 5.2 were purposefully performed using a naïve formulation of the system with components each treated individually in order to test the performance of the proposed algorithm. In practice, systems can be more efficiently represented by grouping subsets of components into “supercomponents” (SCs) in a multi-scale modeling approach (Der Kiureghian and Song 2008). In this section, we discuss the treatment of SCs in the proposed algorithm.

5.3.3.1 Algorithm for supercomponents

For a system of n components, let $\{p_f\}$ = vector of length n defining the failure probabilities of components $C_1, \dots, C_n$. Let n_{SC} denote the number of SCs in the system. Each SC is comprised of a simple sub-system of the overall system, specifically a subset of components that exist either in series or in parallel. We assume that each component of the system can be a member of at most one SC. Additionally, for consistency in the formulation, components that are not in direct series or parallel configuration with other components to form a SC, are treated as SCs on their own. Thus, some SCs may be comprised of only single components. Of course, the greater the number of components that can be grouped into SCs, such that $n_{SC} \ll n$, the greater the efficiency gained when employing the SC formulation. The set of minimum cut sets of the system, $\{MCS\}$, are defined in terms of the SCs.

Let $cCPT_{SC}$ denote the compressed CPT for a generic SC. Because each SC is comprised of either a series or a parallel sub-system of components, $cCPT_{SC}$ can be directly constructed without employing the compression algorithm described in Figure 5.11. Let $\tilde{n}$ denote the number of components comprising the SC. $cCPT_{SC}$ for series and parallel sub-systems consists of two rows and is defined as follows:

$$cCPT_{SC} = \begin{cases} \begin{bmatrix} \text{run} & 0 & 2^{\tilde{n}} - 1 \\ \text{run} & 1 & 1 \end{bmatrix} & \text{for series SC} \\ \begin{bmatrix} \text{run} & 0 & 1 \\ \text{run} & 1 & 2^{\tilde{n}} - 1 \end{bmatrix} & \text{for parallel SC} \end{cases} \quad (5.2)$$

Since $cCPT_{SC}$ is comprised only of runs, the accompanying phrase dictionary is $d_0 = 0$. For n_Q queries, let SC_Q denote the set of SCs which house the queries Q . Also, let $\{p_f|E\}$ = vector of length n_{SC} defining the failure probabilities of SCs $SC_1, \dots, SC_{n_{SC}}$ given the evidence E so that the i th element is $p_{F,i} = \Pr(SC_i = 0|E)$, $i = 1, \dots, n_{SC}$.

To perform inference on a system formulated as SCs, we employ the following algorithm. The mentioned VE inference algorithm is Algorithm B: Inference Algorithm as described in Section 4.3, and the compression algorithm implemented is Algorithm D: Compression Algorithm II as shown in Figure 5.11.

Algorithm E: Algorithm for Supercomponents

Input: $n_{SC}, \{MCS\}, \{p_f\}, Q, E$
Output: $\Pr(Q|E)$

- 1) Determine conditional failure probabilities of SCs for given evidence:
For $i \leftarrow 1$ to n_{SC} , do

Define $cCPT_{SC_i}$ according to Equation (5.2).
Eliminate components in SC_i using VE inference algorithm with:
Input: $n = \tilde{n}_i, \{p_f\}, cCPT_{SC_i}, d_0 = 0, Q = SC_i, E$
Output: $\{p_F|E\}$

end
- 2) Construct $cCPT_{sys}$ using compression algorithm with:
Input: $n = n_{SC}, \{MCS\}$
Output: $cCPT_{sys}, d_0$
- 3) Eliminate SCs using VE inference algorithm with:
Input: $n = n_{SC}, \{p_F|E\}, cCPT_{sys}, d_0, Q = SC_Q, E$
Output: $\{\Pr(SC_Q|E)\}$
- 4) Reorder SCs with query components and perform inference by VE:
For $j \leftarrow 1$ to n_Q , do

For $k \leftarrow SC_Q$ down to 1, do

If $Q_j \in SC_k$
Reorder λ_{k+1} s.t. SC_k is ordered to extreme left, i.e., numbered 1.

Else
Eliminate SCs using VE inference algorithm with:
Input: $n = n_Q, \{\Pr(SC_Q|E)\}, cCPT_{sys}, d_0, SC_j, E$
Output: $\Pr(SC_j|E)$

end

If $Q_j \in SC_j$ where $\tilde{n}_j = 1$

Output: $\Pr(Q_j|E) = \Pr(SC_j|E)$

Else if $Q_j \in SC_j$ where $\tilde{n}_j > 1$

Construct $cCPT_{SC_j}$ according to Equation (5.3).

Eliminate components in SC_j using VE inference algorithm with:

Input: $n = \tilde{n}_j, \{p_f\}, cCPT_{SC_j}, d_0 = 0, Q_j, E$

Output: $\Pr(Q_j|E)$

end

In the construction of $cCPT_{SC_j}$ where $Q \in SC_j$ and $\tilde{n}_j > 1$ in Step 4, we need to “open” the SC to access the components inside the SC to perform inference on the query component(s). To do this, we use the output from Step 3, $\Pr(SC_Q|E)$, which gives us the probability of survival of the overall system given the failure or survival states of the SC s that house query components. Knowing the failure criteria for series and parallel systems, we can directly construct $cCPT_{SC_j}$ for series and parallel SC s as follows:

$$cCPT_{SC_j} = \begin{cases} \begin{bmatrix} \text{run} & \Pr(\text{sys} = 1|SC_j = 0) & 2^{\tilde{n}} - 1 \\ \text{run} & \Pr(\text{sys} = 1|SC_j = 1) & 1 \end{bmatrix} & \text{for series SC} \\ \begin{bmatrix} \text{run} & \Pr(\text{sys} = 1|SC_j = 0) & 1 \\ \text{run} & \Pr(\text{sys} = 1|SC_j = 1) & 2^{\tilde{n}} - 1 \end{bmatrix} & \text{for parallel SC} \end{cases} \quad (5.3)$$

With the compressed CPT, we then eliminate the non-query components in SC to arrive at the posterior probability of the specific query component Q of interest.

5.3.3.2 Example system with supercomponents

We now apply the algorithm for SC s to the example systems described in Section 5.2. Figure 5.14 shows the results in terms of both computation time (left y -axis, solid and dashed lines) and memory storage (right y -axis, circles) for systems of increasing size. Total computation times represent both the time for compression and the time for forward or backward inference. The maximum number of elements that must be stored during both the compression and inference processes are again used as a proxy for the memory storage requirements of the proposed algorithm.

Figure 5.14: Computation times and memory storage requirements for proposed new algorithm with supercomponents implemented

In Figure 5.14, we see that the total number of components in the system is now shown up to 100 components. Looking at the magnitude of the computation time first, the total computation time is on the order of 10^{-2} seconds for a 100-component system. Looking at the trend in computation time as the size of the system increases, it is notably linearly, rather than exponentially, increasing. With regards to memory storage, we see that the maximum number of elements that must be stored remains constant, even as the total number of components in the system increases. Thus, we have achieved significant gains in both memory storage demand and computational efficiency with the proposed algorithm.

5.4 Application: power system

With the objective of reliability assessment of infrastructure systems in mind, we now apply the proposed algorithm to analyze the reliability of the 4-substation power network from Ostrom (2004) shown in Figure 5.15, which was also investigated by Der Kiureghian and Song (2008). The network consists of 3 inputs and 1 output, and 59 components numbered 1-59, where the circles, slashes, and squares represent circuit breakers, switches, and transformers, respectively. Power can flow along any black line, including between substations via the connections shown.

Figure 5.15: Power system application

Additionally, for this system, we implement the method of SCs, representing each triplet of switch-breaker-switch as a SC. Figure 5.16 shows the resulting system, where the dashed squares represent SCs. It is this system of 59 components represented with 20 SCs that we use for analysis.

Figure 5.16: Power system application with supercomponents implemented

5.4.1 Inference

Given an infrastructure system comprised of many interconnected components, as the system shown in Figure 5.15 is, system risk and reliability analyses enable the identification of critical components to support decision making regarding inspection, repair, and replacement. Inference is required to perform these analyses. Figures 5.17 and 5.18 show the results of performing this forward and backward inference, respectively, for the power system application. To facilitate comparison across components, all components are initialized with a prior probability of failure of 0.1.

Figure 5.17: Results of forward inference for power system application

Figure 5.18: Results of backward inference for power system application

Figure 5.17 gives the probability of system failure given component failure for each of the components 1-59. The results of this forward inference support decision making in the management of the power system to minimize the risk of system failure. For example, knowing that probability of system failure is 100% if components 58 or 59 fail clearly indicates the importance of these two components and the need for regular inspection or retrofit of these components to ensure system performance. At a less extreme level, we see that, for example, components 46-48 are more critical than components 10-15. If failed, the updated probability of system failure is 0.425 compared to 0.260.

Figure 5.18 gives the probability of component failure given system failure for each of the components 1-59. The results of this backward inference support decision making in the rehabilitation of the power system to identify what may have led to system failure after the failure event has been observed. For example, the probability that components 58 or 59 have failed has been updated from a prior failure probability of 0.1 to 0.384. In contrast, components 10-15 remain at a probability of failure of 0.1. We see that the evidence on the system state is not informative for these components, and the updated probability of failure equals the prior failure probability.

5.4.1.1 Comparison with component importance measures

The results presented in Figures 5.17 and 5.18 are indicators of component importance in terms of the contribution of individual components to overall system reliability. In this section, we compare the presented inference results with four existing measures of component importance: the Boundary Probability (BP), Risk Achievement Worth (RAW), Risk Reduction Worth (RRW), and the Fussell-Vesely (FV) importance measure. In the following definitions, E_{sys} denotes the event of system failure, and $\Pr(E_{sys})$ denotes the probability of this event occurring. $\Pr(E_{sys+i})$ denotes the probability of system failure with component i replaced by a perfectly reliable component; $\Pr(E_{sys-i})$ denotes the probability of system failure with component i removed, i.e., replaced by a perfectly unreliable component.

The BP, which is also known as Birnbaum importance measure (Cheok et al. 1998), measures the change in the probability of system failure solely due to a change in the component state (Song and Der Kiureghian 2005). It is defined as

$$BP = \Pr(E_{sys-i}) - \Pr(E_{sys+i}) \quad (5.4)$$

The RAW measures the “worth” of component i in “achieving” the current level of risk for the system (Cheok et al. 1998). It quantifies the importance of the component by providing the ratio of the increased probability of system failure when the component is removed compared to the baseline system failure probability, i.e.

$$RAW = \frac{\Pr(E_{sys-i})}{\Pr(E_{sys})} \quad (5.5)$$

The RRW measures the decrease in risk when the component is replaced by a perfectly reliable component (Vinod et al. 2003). This represents the maximum improvement in system reliability that can be achieved from an improvement to the component of interest. RRW is defined as

$$RRW = \frac{\Pr(E_{sys})}{\Pr(E_{sys+i})} \quad (5.6)$$

Finally, the FV importance measure provides the fractional contribution of component i to the overall probability of system failure (Vesely et al., NRC 1983). It is defined as

$$FV = \frac{\Pr(E_{sys}) - \Pr(E_{sys+i})}{\Pr(E_{sys})} \quad (5.7)$$

Table 5.1 gives the values of these four importance measures for the components of the power system application shown in Figure 5.15.

Table 5.1: Component importance measures for power system application

Component #s	BP	RAW	RRW	FV	
10-15	0.000	1.000	1.000	0.000	
25-30					
37-39					
43-45					
55-57					
4-9	0.002	1.006	1.001	6.99×10^{-4}	
16-18					
31-33		1.007		7.86×10^{-4}	
1-3					
49-51	0.019	1.064	1.007	7.11×10^{-3}	
19-21	0.020	1.070	1.008	7.80×10^{-3}	
40-42					
22-24	0.023	1.078	1.009	8.67×10^{-3}	
34-36					
46-48	0.183	1.631	1.075	7.01×10^{-2}	
52-54					
58-59	0.822	3.840	1.461	3.16×10^{-1}	

Comparing Figures 5.17 and 5.18 with Table 5.1, we see that the relative rankings of individual components in terms of their contribution to overall system failure are consistent across all measures. Note that while the varying results for components 1-3 appear only for the RAW and FV measures, these results are also obtained for the other measures, including the posterior probabilities presented in Figures 5.17 and 5.18. The BP, RRW, and posterior probability results for these components do not appear because we have shown results up to 3 decimal places. The inference results presented in the previous section, along with the component importance measures presented in this section, enable us to identify the critical components of a system and inform decision making in the prioritization of limited resources in the management and rehabilitation of systems.

5.4.2 Performance of proposed algorithms

In demonstrating in Section 5.4.1 the ability to perform both forward and backward inference using the proposed algorithm, we also look at the performance of the algorithm in terms of both computational efficiency and memory storage requirements. For the power system shown in 5.15, the time required to compress the system CPT is 12.8 sec. The computation times required for performing forward and backward inference, the results of which are shown in Figures 5.17 and 5.18, are given in Figures 5.19 and 5.20, respectively.

Figure 5.19: Computation times for forward inference for power system application

Figure 5.20: Computation times for backward inference for power system application

From Figure 5.19, the average time required to compute the probability of system failure given component failure is 0.0212 sec. In Figure 5.20, we see an exponential increase in the computation time as the component number increases. This is explained by the reordering of the intermediate factor λ_i that must be performed when arriving at a query node during the VE inference process. Therefore, the heuristic for the inference algorithm proposed in Section 5.3.2 should be implemented whereby queries are ordered to the left in the CPT. Doing so, computation times will be in the lower range, e.g., for components 1-20, when performing backward inference.

Finally, we look at memory storage requirements. The maximum number of elements that must be stored to perform forward and backward inference are 707 and 1059 elements, respectively. While this may not seem particularly small, this is in comparison to a memory storage demand of $2^{59} = 5.8 \times 10^{17}$ elements for a system of $n = 59$ components. Therefore, the ability to construct the BN and perform inference using the proposed algorithms with only 10^3 elements needed represents an orders of magnitude

reduction in memory demand. Thus, the proposed algorithms achieve significant gains in both memory storage and computation time, enabling larger systems to be modeled as BNs for system reliability analysis.

6. Dynamic Bayesian network for structural health monitoring

6.1 Introduction

In the previous chapters, the BN framework was used to enable probabilistic inference for systems modeled as networks of random variables. To perform probabilistic inference for systems that are dynamically evolving, a variation on the BN, the dynamic Bayesian network (DBN), is a powerful tool.

The problem we are interested in is in performing inference on the response of a structure to seismic loading for structural health monitoring applications. An earthquake is a dynamic process, and the analysis of the response of a structure to a seismic excitation requires the analysis of a dynamically evolving system. Therefore, the proposed method is developed based on the DBN framework. The objective is to probabilistically analyze the structural response under seismic loading based on sensor monitoring information. Specifically, to perform probabilistic inference on the relative displacements of the structure over the duration of the excitation based on acceleration measurements.

It is noted that the present analysis assumes linear structural behavior as well as Gaussianity of both the earthquake and ambient vibration input excitations to allow the use of Gaussian models and Kalman filter (KF) described later in this chapter. As such, it is appropriate for serviceability earthquakes and operating-basis seismic events. In this chapter, we first give a brief introduction to DBN, then describe the formulation of the proposed method for dynamic analysis, including the representation of the system as a DBN. The KF approach to process the information in the DBN is then described, which is used to perform probabilistic inference on the dynamically evolving structural response under earthquake loading. Finally, we look at the maximum response in particular, and derive an approximate analytical solution to obtain the distribution of this peak response.

6.2 Formulation of the problem

6.2.1 Background

The following sections describe the proposed method for performing probabilistic inference on the response of the structure to seismic loading based on information from sensors mounted on the structure. A capital bold letter denotes a matrix, such as the mass matrix $\mathbf{M}$, a small bold letter denotes a vector, as in the vector of structural displacements relative to the ground $\mathbf{u}_s(t)$, and a small italic letter denotes a scalar quantity, as in the ground displacement $u_g(t)$. Displacement and acceleration are denoted u and a , respectively, while $\mathbf{z}(t)$ collects displacement and velocity values. Subscripts g and s indicate quantities for the ground and structure, respectively.

We first model the dynamical system as a cascaded system of two dynamical sub-systems: a ground dynamical sub-system and a structural dynamical sub-system as shown in Figure 6.1.

Figure 6.1: Dynamical system model, consisting of ground and structural dynamical sub-systems

The ground dynamical sub-system takes a modulated white-noise input $w(t)$, representing the motion at the bedrock, and outputs the acceleration $a_g(t)$ on the ground surface. The structural dynamical sub-system takes $a_g(t)$ as excitation and produces the structural response $\mathbf{u}_s(t)$, the vector of nodal displacements relative to the ground. Our interest lies not only in the instantaneous values of $\mathbf{u}_s(t)$, but also in its peak values over time.

6.2.2 Ground dynamical sub-system

The equation describing the motion on the ground surface relative to the bedrock is given by

$$\ddot{u}_g + 2\xi_g\omega_g\dot{u}_g + \omega_g^2u_g = -w \quad (6.1)$$

where ω_g and ξ_g define the frequency and damping ratio of the ground filter and w denotes the modulated white-noise acceleration at the bedrock. Written in first-order form with $\mathbf{z}_g = [u_g \quad \dot{u}_g]^T$, Equation (6.1) becomes

$$\dot{\mathbf{z}}_g = \begin{bmatrix} 0 & 1 \\ -\omega_g^2 & -2\xi_g\omega_g \end{bmatrix} \mathbf{z}_g + \begin{bmatrix} 0 \\ -1 \end{bmatrix} w \quad (6.2)$$

The total acceleration at the surface of the ground, a_g , is obtained as

$$a_g = \ddot{u}_g + w = [0 \quad 1]\dot{\mathbf{z}}_g + w = [-\omega_g^2 \quad -2\xi_g\omega_g]\mathbf{z}_g \quad (6.3)$$

6.2.3 Structural dynamical sub-system

The equations of motion for a linear structure subjected to base motion is

$$\mathbf{M}\ddot{\mathbf{u}}_s + \mathbf{C}\dot{\mathbf{u}}_s + \mathbf{K}\mathbf{u}_s = -\mathbf{M}\mathbf{I}a_g + \mathbf{f} \quad (6.4)$$

where $\mathbf{M}$, $\mathbf{C}$, and $\mathbf{K}$ denote the mass, damping, and stiffness matrices, respectively. $\mathbf{I}$ is the influence vector, where each element of $\mathbf{I}$ equals the displacement in the corresponding degree of freedom from the application of a unit ground displacement. $\mathbf{f}$ models a random external force vector that represents ambient vibrations, i.e., random external loads acting on the structure in addition to the earthquake base input, adding uncertainty to the system. In first-order form, using $\mathbf{z}_s^T = [\mathbf{u}_s^T \quad \dot{\mathbf{u}}_s^T]$, Equation (6.4) becomes

$$\dot{\mathbf{z}}_s = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \mathbf{z}_s + \begin{bmatrix} \mathbf{0} \\ -\mathbf{I} \end{bmatrix} a_g + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{f} \quad (6.5)$$

6.2.4 State-space representation

Combining Equations (6.2), (6.3), and (6.5), we obtain a representation of the full dynamical system in first-order form. Defining the system state vector as $\mathbf{z}^T = [\mathbf{z}_g^T \quad \mathbf{z}_s^T]$, we have

$$\dot{\mathbf{z}} = \begin{bmatrix} 0 & 1 & \mathbf{0} & \mathbf{0} \\ -\omega_g^2 & -2\xi_g\omega_g & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{I}\omega_g^2 & \mathbf{I}2\xi_g\omega_g & -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix} \mathbf{z} + \begin{bmatrix} 0 \\ -1 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} w + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{M}^{-1} \end{bmatrix} \mathbf{f} \quad (6.6)$$

Consistent with previous studies (e.g., Gasparini et al. 1983), we define the matrix $\mathbf{A}_c$ and vector $\mathbf{b}_c$ in a state-space representation of the system in continuous time such that $\dot{\mathbf{z}} = \mathbf{A}_c\mathbf{z} + \mathbf{b}_c w + \mathbf{B}_c\mathbf{f}$. Discretizing in time domain in the state-space framework requires the standard transformations $\mathbf{A}_d = e^{\mathbf{A}_c\Delta t}$, $\mathbf{b}_d = \mathbf{A}_c^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{b}_c$, and $\mathbf{B}_d = \mathbf{A}_c^{-1}(\mathbf{A}_d - \mathbf{I})\mathbf{B}_c$ (Johansson et al. 1999). This leads to

$$\mathbf{z}_{k+1} = \mathbf{A}_d\mathbf{z}_k + \mathbf{b}_d w_k + \mathbf{B}_d\mathbf{f}_k \quad (6.7)$$

as the full equation of motion for the system in discrete time step k . For the sake of simplicity, hereafter we drop the subscripts d and write the discretized equation as $\mathbf{z}_{k+1} = \mathbf{A}\mathbf{z}_k + \mathbf{b}w_k + \mathbf{B}\mathbf{f}_k$.

6.2.5 Modeling the seismic excitation

To represent the non-stationarity of the ground motion, we model the acceleration at the bedrock as a modulated, band-limited white-noise process. Thus, $w(t)$ is normally distributed with zero mean and a time-varying variance $\sigma_w^2(t)$. Following Rezaeian and Der Kiureghian (2010), $\sigma_w^2(t)$ is taken as proportional to a gamma probability density function (PDF). The gamma PDF is a reasonable model for this purpose, since it starts

and ends at zero, is zero for negative values, and the shape is skewed with a longer right tail, which is typical of earthquake motions.

The parameters of the gamma PDF are determined in terms of descriptive variables of the seismic event. Specifically, we take the mode of the distribution to coincide with the time of the maximum intensity of the ground motion, t_{eq}^{max} , and the middle 90% of the distribution to represent the effective duration of the earthquake motion, D_{5-95} , which we define as the time between 5% and 95% Arias intensity values. These modeling assumptions lead to the shape and scale parameters of the gamma distribution as $k = \frac{t_{eq}^{max}}{\theta} + 1$

and $\theta = -\frac{1}{2}t_{eq}^{max} + \frac{1}{4}\sqrt{4t_{eq}^{max2} + D_{5-95}^2}$, respectively. The modulating function is scaled by a factor to achieve the desired intensity of the motion. We note that these parameters can be highly uncertain. One approach is to use mean values of the parameters for given earthquake and site characteristics, as described in Rezaeian and Der Kiureghian (2010). Alternatively, these values can be tuned post-event to match the parameters of an observed excitation, where an estimation of the response based on sensor information recorded through the duration of the event is desired. Ideally, these parameters should be estimated from observations of the structural response. However, this would necessitate a costly simulation approach. Hence, as a first attempt at solving this problem, we assume these parameters of the ground motion are known. In Section 7.2.6 in the following chapter, we investigate the robustness of the proposed method to the uncertainty in the input motion parameters. Figure 6.2 shows an example simulation of $w(t)$.

Figure 6.2: Bedrock excitation w with time-varying variance $\sigma_w^2(t)$ proportional to the gamma PDF

This example excitation is a realization of a modulated white-noise process with earthquake parameters set at $t_{eq}^{max} = 20$ sec and $D_{5-95} = 25$ sec, and multiplied by a scale factor of 200. This results in a root mean square of the ground acceleration of 0.64 m/s^2 . For this example, the discretization time step is set as $\Delta t = 0.01$ sec. This effectively cuts frequencies off at 50 Hz (Oppenheim et al. 1999).

6.2.6 System evolution

Given the state-space representation of the system, the system evolution from time step k to $k + 1$ is described as in Equation (6.7). We take $\mathbf{f}_k$ to be normally distributed with zero mean and covariance matrix $\sigma_f^2 \mathbf{I}$, with statistically independent values for different time steps and for different degrees of freedom. Including this additional force increases the uncertainty in our model.

6.2.7 Observation equation

We assume the sensors mounted on the structure measure total acceleration, which is given by

$$\mathbf{a}_t = \ddot{\mathbf{u}}_s + \mathbf{I}a_g \quad (6.8)$$

From Equation (6.4) and $\mathbf{z}$ as defined in Equation (6.6), we obtain

$$\mathbf{a}_t = -\mathbf{M}^{-1}\mathbf{K}\mathbf{u}_s - \mathbf{M}^{-1}\mathbf{C}\dot{\mathbf{u}}_s + \mathbf{M}^{-1}\mathbf{f} = -\mathbf{M}^{-1}[\mathbf{0} \quad \mathbf{0} \quad \mathbf{K} \quad \mathbf{C}]\mathbf{z} + \mathbf{M}^{-1}\mathbf{f} \quad (6.9)$$

Let $\mathbf{S}$ define a matrix that selects the degrees of freedom, where accelerometers are placed. The observation equation at each time step k is then given by

$$\mathbf{y}_k = \mathbf{D}\mathbf{z}_k + \mathbf{v}_k + \mathbf{S}\mathbf{M}^{-1}\mathbf{f}_k \quad (6.10)$$

where $\mathbf{D} = -\mathbf{S}\mathbf{M}^{-1}[\mathbf{0} \quad \mathbf{0} \quad \mathbf{K} \quad \mathbf{C}]$ is the observation matrix and $\mathbf{v}_k$ denotes the vector of measurement errors. We take the measurement errors $\mathbf{v}_k$ to be normally distributed with zero mean and time-independent common variances σ_v^2 , and assume errors at different times and different locations are independent. The objective is to use these sensor measurements to infer the response of the structure as it evolves with time under seismic excitation.

6.3 DBN representation of the system

With a stochastic excitation, a dynamically evolving structural response, and uncertainties in the system, including uncertainties in sensor measurements that will be described in the following chapter, the system to be analyzed is well modeled as a DBN. Figure 6.3 shows the representation of the system as a DBN. As stated previously, in this analysis, we assume a linear Gaussian system to allow the use of the Kalman filter to solve the

DBN. That is, each of the random variables is assumed to be conditionally Gaussian given its parent(s).

Figure 6.3: Representation of the system as a DBN

The DBN consists of a sequence of BNs, each representing the system at a slice in time, $1, \dots, n$. Terms with zero subscripts represent initial values at time zero, which are also uncertain. The DBN shows the evolution of the system state $\mathbf{z}$ over time, taking $\mathbf{w}$ and $\mathbf{f}$ as input. The measurements $\mathbf{y}$ are then taken from $\mathbf{z}$ with measurement error $\mathbf{v}$.

6.4 Kalman filter and Kalman smoother

We use a Kalman filter (KF) approach to process the information in the DBN. This is possible when all the random variables are Gaussian and the relationships between them are linear, which is our case. In addition, we assume our system is observable such that it is possible to infer the internal behavior of the system based on its outputs. Specifically, our objective is to estimate the state of the system $\mathbf{z}$ taking into account measurements $\mathbf{y}$. From Equation (6.7), we predict the mean vector and covariance matrix of the system state at step k as

$$\boldsymbol{\mu}_{\mathbf{z}_k}^P = \mathbf{A}\boldsymbol{\mu}_{\mathbf{z}_{k-1}} \quad (6.11)$$

$$\boldsymbol{\Sigma}_{\mathbf{z}_k\mathbf{z}_k}^P = \mathbf{A}\boldsymbol{\Sigma}_{\mathbf{z}_{k-1}\mathbf{z}_{k-1}}\mathbf{A}^T + \boldsymbol{\Sigma}_{\mathbf{E}_k} \quad (6.12)$$

where $\boldsymbol{\Sigma}_{\mathbf{E}_k} = \mathbf{b}\sigma_{w_k}^2\mathbf{b}^T + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}^2$ and the superscript P denotes the predicted value. We then calculate

$$\mathbf{e}_k = \mathbf{y}_k - \mathbf{D}\boldsymbol{\mu}_{\mathbf{z}_k}^P \quad (6.13)$$

$$\mathbf{R}_k = \mathbf{D}\boldsymbol{\Sigma}_{\mathbf{z}_k\mathbf{z}_k}^P\mathbf{D}^T + \sigma_v^2\mathbf{I} \quad (6.14)$$

$$\mathbf{K}_{G_k} = \Sigma_{Z_k Z_k}^P \mathbf{D}^T \mathbf{R}_k^{-1} \quad (6.15)$$

$\mathbf{e}_k$ is known as the innovation, $\mathbf{R}_k$ characterizes the uncertainty due to the measurement error, and $\mathbf{K}_{G_k}$ is the Kalman gain matrix (Welch and Bishop 1995). The innovation measures the difference between the measurements and the predicted mean values. The Kalman gain takes into account the accuracy of the observations to provide a weight on the information from the measurements compared to the values from the prediction in updating the estimates. Thus, the estimates of the mean vector and covariance matrix of the system state are updated as

$$\mu_{Z_k} = \mu_{Z_k}^P + \mathbf{K}_{G_k} \mathbf{e}_k \quad (6.16)$$

$$\Sigma_{Z_k Z_k} = (\mathbf{I} - \mathbf{K}_{G_k} \mathbf{D}) \Sigma_{Z_k Z_k}^P \quad (6.17)$$

The KF performs a forward pass through the data to update the estimates of the system state as information from measurements becomes available. Once we have information over a fixed time interval, we can perform a backward pass through the data to further update our estimates using the Kalman smoother (KS) (Murphy 2002). To apply the KS, we first compute the smoother gain matrix

$$\mathbf{J}_k = \Sigma_{Z_k Z_k} \mathbf{A}^T (\Sigma_{Z_{k+1} Z_{k+1}}^P)^{-1} \quad (6.18)$$

We then update our estimates:

$$\mu_{Z_k}^{KS} = \mu_{Z_k} + \mathbf{J}_k (\mu_{Z_{k+1}}^{KS} - \mu_{Z_{k+1}}^P) \quad (6.19)$$

$$\Sigma_{Z_k Z_k}^{KS} = \Sigma_{Z_k Z_k} + \mathbf{J}_k (\Sigma_{Z_{k+1} Z_{k+1}}^{KS} - \Sigma_{Z_{k+1} Z_{k+1}}^P) \mathbf{J}_k^T \quad (6.20)$$

It is these final estimates of the system state from applying the KS that we use to probabilistically assess the response of the system under seismic load.

6.5 Distribution of maximum structural response

In reliability analysis, it is often the distribution of extreme values that is critical. Therefore, in this section, we investigate one aspect of the response in particular: the distribution of the maximum structural response over the duration of the seismic excitation. The objective is, for example, to compute the probability of exceedance of a structural response value above a given threshold. In the following section, we derive an analytical solution to compute the probability of a non-zero-mean, nonstationary process exceeding a threshold value.

6.5.1 Analytical solution

We are interested in the probability of the nonstationary process $Z(t)$ exceeding a given threshold ζ during an interval $(0, T)$, where $Z(t)$ is a linear function of the system state $\mathbf{z}_s$ defined in the previous chapter, and T is the duration of the response. We begin with an equation for the probability distribution of the extreme value $Z_{\max} = \max_t Z(t)$ of a nonstationary process:

$$\Pr(Z_{\max} > \zeta) = 1 - \Pr(Z_{\max} \leq \zeta) \cong 1 - \exp \left[- \int_0^T v(\zeta^+, t) dt \right] \quad (6.21)$$

where $v(\zeta^+, t)$ is the mean ζ -level up-crossing rate and T is the duration of the response. The approximation is based on an assumption of Poisson crossings and is valid for high thresholds. Due to conditioning on observed responses, $Z(t)$ is no longer a zero-mean process. Indeed, the mean of $Z(t)$ varies with time. Thus, to obtain $v(\zeta^+, t)$, we cannot use the well-known formula for up-crossings of a Gaussian process with a constant mean above a fixed threshold (Rice 1944).

Instead, we define the process $X(t) = Z(t) - \mu_Z(t)$, where $X(t)$ is a zero-mean process with $\sigma_X(t) = \sigma_Z(t)$. The up-crossings of the non-zero-mean process $Z(t)$ above a fixed threshold ζ are identical to the up-crossings of the zero-mean process $X(t)$ above a time-varying threshold $\eta(t) = \zeta - \mu_Z(t)$. Consider the sketch shown in Figure 6.4. We are interested in deriving the mean up-crossing rate of $X(t)$ above $\eta(t)$.

Figure 6.4: Up-crossing of a non-stationary zero-mean process above a time-varying threshold

From Figure 6.4, we see that during the small time interval $(t, t + \Delta t]$, the probability of encountering an up-crossing of level $\eta(t)$ is

$$\begin{aligned} & \Pr[\{X(t) < \eta(t)\} \cap \{\dot{\eta}(t) < \dot{X}(t)\} \cap \{\eta(t) + \dot{\eta}(t)\Delta t < X(t) + \dot{X}(t)\Delta t\}] \\ &= \Pr\left[\left\{\eta(t) + (\dot{\eta}(t) - \dot{X}(t))\Delta t < X(t) < \eta(t)\right\} \cap \{\dot{\eta}(t) < \dot{X}(t)\}\right] \\ &= \int_{\dot{\eta}}^{\infty} \int_{\eta + (\dot{\eta} - \dot{x})\Delta t}^{\eta} f_{X\dot{X}}(x, \dot{x}, t) dx d\dot{x} \\ &= \Delta t \int_{\dot{\eta}}^{\infty} (\dot{x} - \dot{\eta}) f_{X\dot{X}}(\eta, \dot{x}, t) d\dot{x} \end{aligned} \quad (6.22)$$

Since in a small interval, the probability of more than one up-crossing is of higher order, we obtain

$$v(\zeta^+, t) = v_X(\eta(t)^+, t) = \int_{\dot{\eta}}^{\infty} (\dot{x} - \dot{\eta}) f_{X\dot{X}}(\eta, \dot{x}, t) d\dot{x} \quad (6.23)$$

In our case,

$$f_{X\dot{X}}(\eta, \dot{x}, t) = \frac{1}{2\pi\sigma_X\sigma_{\dot{X}}\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{\eta^2}{\sigma_X^2} - 2\rho\frac{\eta\dot{x}}{\sigma_X\sigma_{\dot{X}}} + \frac{\dot{x}^2}{\sigma_{\dot{X}}^2}\right)\right] \quad (6.24)$$

wherein $\eta(t)$, $\sigma_X(t)$, $\sigma_{\dot{X}}(t)$ and $\rho(t) = \rho_{\dot{X}(t)X(t)}$ are all functions of time. Using Equation (6.24) in Equation (6.23), we have

$$v_X(\eta(t)^+, t) = \frac{1}{2\pi\sigma_X\sigma_{\dot{X}}\sqrt{1-\rho^2}} \left\{ \int_{\dot{\eta}}^{\infty} \dot{x} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{\eta^2}{\sigma_X^2} - 2\rho\frac{\eta\dot{x}}{\sigma_X\sigma_{\dot{X}}} + \frac{\dot{x}^2}{\sigma_{\dot{X}}^2}\right)\right] d\dot{x} - \dot{\eta} \int_{\dot{\eta}}^{\infty} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{\eta^2}{\sigma_X^2} - 2\rho\frac{\eta\dot{x}}{\sigma_X\sigma_{\dot{X}}} + \frac{\dot{x}^2}{\sigma_{\dot{X}}^2}\right)\right] d\dot{x} \right\} \quad (6.25)$$

Evaluation of the integral leads to

$$v_X(\eta(t)^+, t) = \frac{\exp\left[-\frac{1}{2}\frac{\eta^2}{\sigma_X^2}\right]}{2\pi\sigma_X\sigma_{\dot{X}}\sqrt{1-\rho^2}} \left\{ \sigma_{\dot{X}}^2(1-\rho^2) \exp\left[-\frac{r^2}{2\sigma_X^2(1-\rho^2)}\right] + \sqrt{2\pi(1-\rho^2)}\sigma_{\dot{X}} \left[1 - \Phi\left(\frac{r}{\sigma_X\sqrt{1-\rho^2}}\right)\right] \left(\frac{\sigma_{\dot{X}}\rho\eta}{\sigma_X} - \dot{\eta}\right) \right\} \quad (6.26)$$

where $r = \dot{\eta} - \frac{\rho\eta\sigma_{\dot{X}}}{\sigma_X}$ and $\Phi(\cdot)$ indicates the normal CDF. The values $\sigma_X(t)$, $\sigma_{\dot{X}}(t)$ and $\rho(t) = \rho_{\dot{X}(t)X(t)}$ are outputs from the DBN analysis. Using Equations (6.21) and (6.26), we obtain an approximation of the probability of exceedance of the maximum of the process $Z(t)$ above a threshold ζ , $\Pr(Z_{max} > \zeta)$. This is the complementary cumulative distribution function (CDF) of the maximum response Z_{max} as a function of threshold value, ζ .

The following chapter applies the DBN framework described in this chapter as well as the approximate analytical solution derived above for obtaining the distribution of the maximum structural response to a multi-story shear-type building application to illustrate the effectiveness of these methods.

7. Application and robustness investigation of proposed DBN framework

7.1 Introduction

In this chapter, we apply the proposed DBN framework described in the previous chapter to a multi-story, shear-type building subjected to earthquake ground motion. We are interested in probabilistically inferring the structural responses of this building to seismic load based on measurements from one or several sensors mounted on the structure. We first present the Kalman filter (KF) and Kalman smoother (KS) estimates of a selected inter-story drift response and show that the latter produces more accurate estimates. We then compare the results from the approximate analytical solution for the distribution of the maximum response to results obtained from Monte Carlo (MC) simulations. We show that the analytical solution is good for high thresholds. Next, we investigate the influence of the sensor placement on the accuracy of the estimation. Finally, we investigate the effect of uncertainties in the system parameters, i.e., stiffness, mass, and damping values of the structure, as well as uncertainties in the characterization of the stochastic ground motion on the estimation of the system state. Robustness of the response estimates to these uncertainties is demonstrated.

7.2 Application

We apply the proposed method to a 10-story, shear-type building as shown in Figure 7.1.

Figure 7.1: 10-story shear-type building model

The building is of nominally uniform mass and stiffness with these parameter values set such that the nominal fundamental period is 1 sec. We assume the building is modally damped with nominal damping ratios being 5% in each mode. The parameters of the ground filter are set at $\omega_g/2\pi = 1.5$ Hz and $\xi_g = 0.4$. We take the stochastic excitation at the bedrock level as described for the example realization in Figure 6.3, with earthquake parameters $t_{eq}^{max} = 20$ sec and $D_{5-95} = 25$ sec. The estimated mean peak ground acceleration is 1.6 m/s^2 , obtained by multiplying the root mean square of the ground acceleration by a peak factor of 2.5.

We use simulation to investigate the accuracy of the analysis method. We first simulate a seismic event as well as the ambient noise. We compute the structural response from this generated combined excitation, and we call this the “actual” response. We then simulate measurements of this response, including measurement noise. These represent the observations of floor accelerations that we obtain from the accelerometers mounted on the structure. Then, assuming we have only these noisy measurements of floor accelerations and the system and ground motion parameters, we use our dynamic Bayesian network (DBN) formulation of the system to estimate the response of the structure to the stochastic seismic loading.

The proposed formulation enables us to perform probabilistic inference on the relative displacements of the structure from our estimate of the system state $\mathbf{z}_s$. In this study, we are interested in the inter-story drift response in particular. While we are able to analyze the inter-story drift throughout the building, for consistency in the results presented in the following sections when looking at time trajectories of the response, we will be examining one inter-story drift in particular, inter-story drift #5 between floors 4 and 5.

7.2.1 Estimates of the response

With one sensor placed on the top floor of the building, Figure 7.2 shows the time history of the inter-story drift #5, including the “actual” value and the Kalman filter (KF) and Kalman smoother (KS) estimates. The standard deviation of measurement error is set at $\sigma_v = 0.5 \text{ m/s}^2$.

**Figure 7.2: Time history of inter-story drift #5:
actual response, KF, and KS estimates**

Because the time histories are close, Figure 7.3 shows the results for one particular peak that occurs at $t = 19 \text{ sec}$. Any other segment of the time history can similarly be chosen to analyze the results of the estimation. For consistency, hereafter, we will examine the segment of the time history around the peak at the selected time $t = 19 \text{ sec}$.

Figure 7.3: Time history of inter-story drift #5: actual response, KF, and KS estimates (peak at $t = 19$ sec)

In Figure 7.3, for the KF and KS estimates, the bolded lines indicate the mean estimates while the thinner lines indicate the mean estimates $\pm$ one standard deviation. Comparing the actual response with the estimates, we see that, consistent with the theory, employing the KS improves the accuracy of the response estimate compared to using only the KF. Utilizing the additional information of the measurements over the entire time history results in a KS mean estimate that is closer to the actual inter-story drift values and a decreased variance in the estimate. Thus, hereafter, only the KS results will be used.

7.2.2 Distribution of maximum response

In analyzing the distribution of the maximum response, we compare the results we obtain from the analytical solution presented in Section 6.6.1 with those obtained from MC simulations. For MC, we take a fixed sensor recording and then generate multiple realizations of the posterior process consistent with the inference on the DBN. These are essentially realizations of the conditioned response process for the given observed sensor recordings. Each MC realization produces a time history of the evolution of the structural response, including the inter-story drift. Sample trajectories are shown in Figure 7.4 for 10 randomly selected realizations of inter-story drift #5, plus the “actual” value and the KS estimate.

Figure 7.4: MC realizations of inter-story drift #5 compared to actual response and KS estimate (peak at $t = 19$ sec)

In Figure 7.4, we see that the MC realizations largely lie within the mean KS estimate of the response $\pm$ two standard deviations. This represents the 95% confidence interval. Having these MC samples, we then take, for each realization, the maximum of the inter-story drift over the time period of analysis to obtain distributions of the maximum response. Figure 7.5 shows the normalized frequency diagrams of the maximum inter-story drifts for all 10 stories of the structure.

Figure 7.5: Normalized frequency diagrams of maximum inter-story drift from MC simulations for all 10 stories of the structure

The distributions shown in Figure 7.5 enable us to obtain the statistics of the maximum inter-story drift for each story of the structure. We note that the dispersions in these distributions are rather small in spite of the stochastic nature of the excitation. This is due to the conditioning on the measured acceleration response.

The analytical solution derived in Section 6.6.1 gives us the probability of exceedance of the maximum inter-story drift above a certain threshold. For comparison, we also compute these probabilities from the MC simulations. Figure 7.6 shows the complementary CDF's of the inter-story drifts for all 10 stories obtained from the analytical solution versus MC. The analytical results (indicated by the solid lines) are obtained by using Equations (6.21) and (6.25) in Chapter 6. The MC results (indicated by the dotted lines) are the empirical CDF's.

Figure 7.6: Complementary CDF of maximum inter-story drift from analytical solution and MC simulations

In Figure 7.6, the probabilities of exceedance on the vertical axis are on a logarithmic scale, and the threshold values are given on the horizontal axis. Each curve gives the probability of the maximum inter-story drift exceeding the threshold value as a function of the threshold value. Looking at the analytical results, we see a phenomenon that we know cannot be true, namely a non-monotonically decreasing complementary CDF for the probability that the maximum inter-story drift exceeds the threshold value, at high probabilities of exceedance. This is due to the approximation made in Equation (6.21) for the extreme values of a nonstationary process. The approximation assumes Poisson crossings of the process above the threshold. However, at lower thresholds, there may be clusters of dependent crossings. Furthermore, once a threshold is crossed, the process cannot make another crossing as long as it stays above that threshold. This “transition time” effect, which can be significant at low thresholds, is not accounted for in the Poisson model. This result shows that the Poisson crossings assumption in analyzing extreme values is indeed only valid for high thresholds and low probabilities of exceedance.

Indeed, looking at the results in Figure 7.6, we see that the anomalies in the analytical solution all occur at probabilities of exceedance above 0.25. We focus instead at high thresholds and low probabilities of exceedance, which is where damage occurs and the range we are interested in for our analysis of the structural response. Comparing the results obtained from the analytical solution versus MC at these high thresholds, we see close agreement between the complementary CDF’s at probabilities of exceedance below 0.1 for all inter-story drifts. In addition, the analytical solution is able to compute results for small probabilities of exceedance that the empirical CDF from MC simulation is not

able to capture due to the finite number of MC samples. Thus, our methodology enables us to estimate the probability that the maximum response will exceed a high threshold, and the analytical solution we have derived enables us to obtain these probabilities without needing to perform simulations.

7.2.3 Varying measurement characteristics

We now investigate the effect of varying sensor measurement characteristics on our ability to perform the estimation. We vary the sensor number, placement, and precision. Figure 7.7 shows the effect of varying sensor configuration on the structure, comparing the actual inter-story drift #5 response with the mean KS estimates from: one sensor placed on the bottom of the structure at floor 1, one sensor placed at the top of the structure at floor 10, and four sensors placed on the structure one each at floors 1, 4, 7, and 10.

Figure 7.7: Mean KS estimates of inter-story drift #5 for varying sensor number and placement (peak at $t = 19$ sec)

Figure 7.7 shows that, as expected, more information results in more accurate estimates of the response. The mean estimate of inter-story drift utilizing the measurements from four sensors placed on the structure is more accurate than the estimate that utilizes measurements from only one sensor. However, looking at the results of using only one sensor, we see that the placement of the sensor also matters. Because we are looking at the inter-story drift at floor 5 where no direct measurements are being made, the sensors at the bottom and top of the structure on floors 1 and 10, respectively, are equally far from the response of interest. The accuracy of the corresponding drift estimates, however, are not equal. The mean estimate using the information from the top sensor is far more accurate than the mean estimate using the information from the bottom sensor. In addition, Figure

7.8 shows the standard deviations of the KS estimates over the entire duration of the loading using the same sensor configurations.

Figure 7.8: Standard deviations of KS-estimated inter-story drift #5 for varying sensor number and placement

Corresponding to the results shown in Figure 7.7, in Figure 7.8, the standard deviations are lowest for the estimate using information from all four sensors. Comparing the effect of differing placement for one sensor, the standard deviations are much higher for the estimate that uses the sensor at the bottom floor compared to the estimate that uses the sensor at the top floor. From both Figures 7.7 and 7.8, we see that an accelerometer placed at the upper floor of a building instead of the lower floor leads to more accurate estimates of the mean and a smaller variance.

This result is explained by considering the signal-to-noise ratio (SNR). The amplitude of the floor acceleration in response to a seismic excitation is higher at the higher floors, so with the same device, the SNR is higher for measurements at the higher floors. Therefore, if given the choice, one would prefer to place accelerometers at the upper floors of a building instead of the lower floors, to obtain measurements that are more informative regarding the structural response and will result in more accurate estimates of the system state.

Next, we vary the sensor precision. Figure 7.9 shows the actual inter-story drift #5 response compared to the mean KS estimates based on acceleration measurements made with 0.5 m/s^2 , 1 m/s^2 and 2 m/s^2 error standard deviations. The measurement information is from one sensor placed at the top of the structure at floor 10. In addition, Figure 7.10

shows the standard deviations of the KS estimates over the entire duration of the loading using of the same three levels of precision as in Figure 7.9.

Figure 7.9: Mean KS estimates of inter-story drift #5 for varying sensor precision (peak at $t = 19$ sec)

Figure 7.10: Standard deviations of KS-estimated inter-story drift #5 for varying sensor precision

Figure 7.9 shows that more precise sensors result in more accurate estimates of the mean system response, and Figure 7.10 shows that increasing the precision of the sensors reduces the variance of the estimate, although the effect is not large. Improving the sensor by 4 times, i.e., reducing the error in the measurement from 2 m/s^2 to 0.5 m/s^2 , only reduces the average standard deviation of the estimate by about 30%, from $12 \times 10^{-4} \text{ m}$ to $8.5 \times 10^{-4} \text{ m}$. In contrast, reexamining Figure 7.8, we see that changing the placement of the same single sensor from the bottom of the structure to the top of the structure reduces the average standard deviation of the estimate by more than half, from $18 \times 10^{-4} \text{ m}$ to $8.5 \times 10^{-4} \text{ m}$.

Finally, to compare the accuracy of the estimates across different combinations of sensor number, placement, and precision, Figure 7.11 shows the root mean square errors (RMSE's) of the mean KS estimates of the inter-story drift #5 using one sensor at the top floor compared to four sensors at floors 1, 4, 7, and 10, and with varying precisions of 0.5 m/s^2 , 1 m/s^2 , and 2 m/s^2 error. The MSE is the average squared error between the estimated and actual inter-story drift values computed across the duration of the response, then the square root taken to result in the RMSE, which has the same units as the quantity of interest. For the results shown in Figure 7.11, placing one sensor at the bottom floor results in a RMSE of 0.0018 m , significantly higher in magnitude than the results for the sensor at the top floor and for the four sensors, and therefore is not shown.

Figure 7.11: RMSE's of KS-estimated inter-story drift #5 varying sensor number, placement, and precision

Figure 7.11 shows that keeping sensor precision constant, placing four sensors on the building compared to placing one on the top floor reduces the RMSE of the estimate by 27% for the 2 m/s^2 error sensor, 32% for the 1 m/s^2 error sensor, and 37% for the 0.5 m/s^2 error sensor. In addition, the RMSE of mounting four sensors of lesser quality throughout the building is larger compared to mounting one sensor of higher quality at the top floor of the building. In the following section, measurement information from one sensor mounted at the top floor of the building that makes measurements with 0.5 m/s^2 error is used for the analysis.

Lastly, when comparing the two monitoring strategies shown in Figure 7.11, we note that the reliability of the structural health monitoring device is an additional consideration to the accuracy of the estimation of the system state. Assuming equal device reliability, a monitoring system consisting of four sensors will be more robust to sensor device failure than a system comprised of only one sensor.

7.2.4 Robustness to uncertainty in structural parameters

In the previous analyses, we assumed that parameters of the structure were known. In reality, these parameters are subject to uncertainty. In this section, we investigate the robustness of the estimation results to this uncertainty in the structural parameters.

7.2.4.1 Uncertainty in stiffness

Figure 7.12 shows the effect of uncertainty in the story stiffnesses of the structure on the accuracy of the estimation of the inter-story drift #5 response over the full time history. The figure shows the “actual” response and the KS estimate for increasing values of the coefficients of variation (c.o.v.) of story stiffnesses. Because the time histories are close, Figure 7.13 shows the results for the peaks that occur between $t \in [19, 24]$ sec. Any other segment of the time history can similarly be chosen to analyze the results of the estimation. For consistency and clarity, hereafter, we examine the peaks in this time segment in our analysis.

Figure 7.12: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of story stiffnesses 0-20%

Figure 7.13: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of story stiffnesses 0-20% (peaks in interval $t = [19, 24]$ sec)

In this analysis, we use the nominal values of the story stiffnesses to obtain the KS estimate of the response. The nominal values represent our best guess of the structural parameters, e.g., based on design drawings. However, for investigating the robustness of the inference method, we simulate synthetic data using a system different from the one assumed in the data analysis. For this “actual” structure, the story stiffnesses are considered as random variables and vary from the nominal values. We sample the “actual” stiffness of each story randomly from the lognormal distribution with the mean equal to the nominal value and over a range of c.o.v.'s, assuming statistical independence from

story to story. It is these randomly sampled values that we use in our simulation of the actual response and measured values.

In Figures 7.12 and 7.13, the c.o.v.'s of the actual stiffnesses range from 0-20%. Because each level of c.o.v. produces a different trajectory for the actual response, we compare each estimate with the corresponding actual response for that c.o.v. value, as shown in each subplot. We see that, as expected, as the c.o.v. increases, the KS estimates slightly diverge from the actual responses. The effect is slight, however. Even as c.o.v. increases up to 20%, we see close correspondence between the actual and estimated inter-story drift responses.

To quantify the effect of uncertainty in the structural parameters on our inference results, we examine the difference between the actual and estimated responses as a function of the c.o.v.'s of story stiffnesses. Figure 7.14 shows the RMSE's of the mean KS estimates of inter-story drift #5 for varying c.o.v.'s. The MSE is computed over the duration of the response as the average squared error between the estimated and actual inter-story drift values, and the RMSE is its square root. For Figure 7.14, the RMSE's were computed across the range of c.o.v.'s with a step size of 0.01. The circles indicate the values at the 5 c.o.v. levels presented previously, i.e., for c.o.v. = 0%, 5%, 10%, 15%, and 20%.

Figure 7.14: RMSE's of KS-estimated inter-story drift #5 with varying c.o.v.'s of story stiffnesses 0-20%

In Figure 7.14, we see that, as we expect, the RMSE increases with the increasing uncertainty in the story stiffnesses. The increase, however, is gradual and relatively small in magnitude. Specifically, the RMSE increases from 7.6×10^{-4} m to 1.4×10^{-3} m over c.o.v.'s ranging from 0-20%, with the maximum RMSE an order of magnitude smaller than the peak inter-story drift response. Note that the non-monotonicity of the RMSE is due to random variation, as the RMSE shown is the result from one simulation at each value of c.o.v.

The reason the inference is robust relative to the uncertainty in the structural stiffnesses is due to the updating that is performed with the DBN formulation of the problem. The variability in the structural parameters affects the estimate of the system state. However, at each time step, we have information from the sensor measurements, and we use this information to update the estimation. Because this updating occurs at every time step, in this case $\Delta t = 0.01$ sec, the estimate is quickly corrected by the measurement information before the estimate has time to evolve incorrectly based on the incorrectly assumed nominal values for structural parameters in the estimation equations.

Finally, in looking at inter-story drift, one possible explanation for the robustness of the results to uncertainty in the structural parameters is that there may be a cancellation of errors that occurs when subtracting displacements of adjacent floors to compute the inter-story drift. To ensure that the accuracy of the estimation is not due to this cancellation, we also investigated the results in estimating individual floor displacements. The results of this analysis, not shown here, showed similar robustness to model uncertainties.

7.2.4.2 Uncertainty in stiffness and mass

Now, we add to the uncertainty in the structural parameters, varying both stiffness and mass. We assume nominal values for the story stiffnesses and masses that we use in our estimation of the system state. The “actual” stiffness and mass values that we use to simulate the actual structural response are both randomly sampled from the lognormal distribution, with the means equal to the nominal values and c.o.v.’s ranging from 0-20%. Figure 7.15 shows the time history of the response and the estimation of inter-story drift #5 for the peaks that occur in the time interval $t \in [19, 24]$ sec.

Figure 7.15: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of story stiffnesses and masses 0-20% (peaks in interval $t = [19, 24]$ sec)

In Figure 7.15, we see that, as expected, the estimation performs best for low c.o.v.'s. However, even as the c.o.v.'s for both stiffness and mass parameters increase up to 20%, the actual and KS-estimated values of the inter-story drift response display close correspondence. Figure 7.16 quantifies the robustness of the inference to uncertainty in the structural parameters by plotting the RMSE's of the mean KS estimates of inter-story drift #5 computed over the duration of the response as a function of c.o.v. The c.o.v.'s for both story stiffnesses and masses range from 0-20%.

Figure 7.16: RMSE's of KS-estimated inter-story drift #5 with varying c.o.v.'s of story stiffnesses and masses 0-20%

In Figure 7.16, we see that, as expected, the RMSE's do increase with increased uncertainty in the story stiffnesses and masses. Similar to the result in Figure 7.14, however, the increase is relatively small in magnitude. Specifically, the RMSE increases from 7.6×10^{-4} m to 2.6×10^{-3} m over the range of c.o.v.'s for the structural parameters from 0-20%. Even the maximum RMSE, which displays a large increase due to random variation at c.o.v. = 20%, is an order of magnitude smaller than the peak inter-story drift response with both stiffness and mass parameters subject to uncertainty.

7.2.4.3 Uncertainty in stiffness, mass, and damping

Finally, we allow all three structural parameters – stiffness, mass, and damping – to vary randomly. We assume nominal values for the parameters that we use in our estimation of the system state, while the “actual” parameter values are randomly sampled from the lognormal distributions. The distribution means for all three parameters are equal to the nominal values, and the c.o.v.'s for stiffness and mass range from 0-20%. Because it is more difficult to estimate damping, we increase the uncertainty in our randomized samples of the damping ratio to c.o.v.'s ranging from 0-40%. Figure 7.17 shows the time history of the actual response of the inter-story drift #5 and the KS estimates of the response for the peaks that occur in the time interval $t \in [19, 24]$ sec. Figure 7.18 plots the RMSE's of the mean KS estimates of inter-story drift #5 computed over the duration of the response as a function of the c.o.v.'s of the structural parameters.

Figure 7.17: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of story stiffnesses and masses 0-20%, of damping ratios 0-40% (peaks in interval $t = [19, 24]$ sec)

Figure 7.18: RMSE's of KS-estimated inter-story drift #5 with varying c.o.v.'s of story stiffnesses and masses 0-20%, of damping ratios 0-40%

Looking at Figures 7.17 and 7.18, we see that the estimation is robust to uncertainty in the structural parameters. Even with all three structural parameters subject to random variation, the KS estimate closely corresponds with the actual response. In addition, while the RMSE's do increase with increasing c.o.v. values, the change in the RMSE from 7.8×10^{-4} m to 8.5×10^{-4} m (a 10% increase) with a maximum of 1.9×10^{-3} m over the range of c.o.v.'s is gradual and relatively small in magnitude.

7.2.5 Robustness to uncertainty in ground parameters

In this section, we investigate the robustness of the inference results to uncertainty in the ground parameters. We assume nominal values for the ground parameters to be $\omega_g/2\pi = 1.5$ Hz and $\xi_g = 0.4$. These represent our best guess of the soil characteristics and are the values we use in our estimation of the system state. In reality, the ground parameters are subject to random variation. The “actual” parameter values that we use to simulate the actual structural response are randomly sampled from the lognormal distribution with the mean equal to the nominal value and over a range of c.o.v.'s from 0-20%. Figure 7.19 shows the full time history of the actual and estimated inter-story drift #5 response. Figure 7.20 shows the results for the peaks that occur between $t \in [19, 24]$ sec.

Figure 7.19: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of ground parameters 0-20%

Figure 7.20: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of ground parameters 0-20% (peaks in interval $t = [19, 24]$ sec)

In Figures 7.19 and 7.20, we see that varying the ground parameters produces varying structural response trajectories. However, we see that in each case, the KS estimate of the response is close to the actual response. Despite the variation in the ground parameters and response trajectories, the estimation performs well. Figure 7.21 shows the RMSE's of the mean KS estimates of inter-story drift #5 computed over the duration of the response as a function of c.o.v. The c.o.v.'s for the ground parameters range from 0-20%.

Figure 7.21: RMSE's of KS-estimated inter-story drift #5 with varying c.o.v.'s of ground parameters 0-20%

In Figure 7.21, we see that the accuracy of the estimation appears to be insensitive to the degree of uncertainty in the ground parameters. While the RMSE of the estimation is subject to random variation, it remains essentially constant as uncertainty in the ground parameters increases, with a mean RMSE of 7.7×10^{-4} m over the range of c.o.v.'s from 0-20%. The maximum value of the RMSE is around 10% of the maximum RMS of the inter-story drift.

7.2.6 Robustness to uncertainty in input motion parameters

Finally, we investigate the robustness of the proposed method to uncertainty in the input motion parameters. We vary the descriptive variables of the seismic event, i.e., the time of the maximum intensity of the ground motion, t_{eq}^{max} , and the effective duration of the earthquake motion, D_{5-95} . In addition, we vary the scale factor of the gamma modulating function, i.e., to randomize the intensity of the earthquake. We assume the nominal values $t_{eq}^{max} = 20$ sec, $D_{5-95} = 25$ sec, and a scale factor of 200 in our estimation of the system state. In reality, these ground motion parameters are highly uncertain. We randomly sample from the lognormal distribution with the mean equal to the nominal value and over a range of c.o.v.'s from 0-40% to obtain the “actual” parameter values that we use to simulate the actual ground motion. With this procedure, Figure 7.22 shows the full time history of the actual and estimated inter-story drift #5 response. Figure 7.23 shows the results for the peaks that occur between $t \in [19, 24]$ sec.

Figure 7.22: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of input motion parameters 0-40%

Figure 7.23: Time history of inter-story drift #5: actual response versus KS estimates with varying c.o.v.'s of input motion parameters 0-40% (peaks in interval $t = [19, 24]$ sec)

In Figures 7.22 and 7.23, we see the effect of varying input motion parameters on the trajectories of the structural response. However, we see that in each case, the KS estimate of the response is close to the actual response, and the estimation performs well across time histories. Figure 7.24 shows the RMSE's of the mean KS estimates of inter-story drift #5 computed over the duration of the response as a function of c.o.v. The c.o.v.'s for the input motion parameters range from 0-40%.

Figure 7.24: RMSE's of KS-estimated inter-story drift #5 with varying c.o.v.'s of input motion parameters 0-40%

In Figure 7.24, we see that the variability in the RMSE of the estimate increases slightly with increasing c.o.v. of the input motion parameters. The value of the RMSE, however, remains essentially constant over the range of c.o.v.'s 0-40% with a mean of 7.9×10^{-4} m.

In this chapter, we have shown our DBN formulation of the problem of estimating the state of a system subject to stochastic excitation based on sensor measurements to be robust to uncertainty in the structural parameters, the ground parameters, as well as the input motion parameters. In the proposed methodology, our estimate of the system state is updated at every time step with information from sensor measurements. This enables us to minimize the effect of uncertainty in the system parameters on the accuracy of the estimation. Thus, we see the importance of sensor measurements in our ability to perform inference on the dynamically evolving system under seismic load. From our analyses varying measurement characteristics, we see that for structural health monitoring applications, the placement of the sensors has a more significant effect than improving sensor precision, and we have shown the proposed method to perform well even under conditions of uncertainty.

8. Summary, Conclusions, and Future Work

8.1 Summary of major contributions

This study presents novel Bayesian network (BN) methodologies for modeling and reliability assessment of infrastructure systems. We propose a compression algorithm that significantly reduces the memory storage requirements for the BN modeling of systems. The algorithm compresses the conditional probability table (CPT) associated with the system node in the BN as a combination of runs and phrases. The sizes of the compressed CPT and the associated phrase dictionary are typically orders of magnitude smaller than the size of the original CPT.

We also propose an inference algorithm that is able to perform both backward and forward inference on the compressed matrices. The algorithm utilizes variable elimination to perform exact inference, where the intermediate factors created after each elimination step are also stored in compressed form. No decompression or recompression of these factors during the inference calculations is necessary. This enables the savings in memory storage achieved by implementing the compression algorithm to be preserved through the inference process.

The gains achieved by the compression algorithm in memory storage are accompanied by a trade-off in computation time. Several heuristics, including efficient component ordering and a multi-scale modeling approach employing supercomponents, are presented to improve computational efficiency. Through the application of the developed algorithms and heuristics to example systems, we demonstrate the ability to model systems of increasing size with orders of magnitude savings in both memory storage and computation time. Together, these algorithms enable larger systems to be modeled as BNs for system reliability analysis.

In addition to reliability assessment at the system level, we present a dynamic BN (DBN) framework to analyze reliability at the structural level. The proposed DBN methodology enables inference on the response of a structure as it dynamically evolves in time due to an excitation that is stochastic in nature, e.g., an earthquake ground motion, and based on

sensor measurements that are uncertain. We look at the maximum response in particular, and derive an approximate analytical solution for estimating the distribution of the peak response. This solution takes as input values obtained from the DBN analysis to compute the probability of the response exceeding a given threshold over the duration of the response.

In applying the proposed DBN framework to a multi-story shear-type building, we demonstrate the ability of the methodology to accurately estimate the structural response, e.g., inter-story drift, based on sensor information as the building is subjected to a seismic hazard. We investigate the effect of varying measurement characteristics and find that sensor placement is more significant than sensor precision. Finally, we show the method to be robust to uncertainties in the system, including in the structural parameters, the ground parameters, as well as the input bedrock motion parameters.

8.2 Future work

The work presented in this study can be extended in several ways. Areas for further study include:

- Extension to multi-state components and systems for more detailed analysis of flow networks, including transportation and water distribution systems. As described previously, while the values in the system CPT for a binary system are given in a vector of 0 and 1 binary values, for a multi-state flow system, the values in the CPT are given in a matrix with the number of columns equal to the number of system states. This matrix, though, is still comprised solely of 0s and 1s. Thus, the number of values in the CPT remains finite, and the CPT for multi-state systems can be compressed using an extension of the compression algorithm described in this study.
- Extension to multi-hazard assessment to perform system reliability analysis under multiple hazards. Combining the presented BN methodology with various hazard models would enable detailed post-hazard risk assessment and management of infrastructure systems based on varying hazard scenarios. As the study of infrastructure resilience increases in importance as our critical infrastructure continues to age, the ability of the BN to probabilistically model a system and update the assessment of the system as new information becomes available or for prescribed scenario events makes it a powerful tool for system reliability analysis to improve the resilience of our critical infrastructure.
- Integration of decision and utility nodes into the BN model to create influence diagrams (IDs). The ID is a probabilistic network used to support decision making under conditions of uncertainty. A decision node is defined by a finite set of states that correspond with the possible decision alternatives. A utility node gives the value of utility as a function of the states of its parents. For each decision alternative, the expected utility is calculated, and the alternative selected that maximizes

the expected utility. Extending the BN model to include the ID will allow development of a decision-support system that can address such issues as decision making for inspection, repair, or retrofit, or design of infrastructure systems that are subject to natural or man-made hazards.

- Extension of the DBN methodology for structural health monitoring to nonlinear structural behavior. For nonlinear systems, the standard Kalman filter approach for processing the information in the DBN can no longer be used. Instead, methods for nonlinear estimation, including the extended Kalman filter and the unscented Kalman filter will be investigated. Employing these methods will extend the use of the DBN framework beyond serviceability earthquakes and operating-basis seismic events to perform inference on structural behavior under more extreme hazard events.

Infrastructure systems are complex. In an environment of limited resources, system reliability analysis is essential to identify the critical components of a system to make decisions to inspect, repair, or replace. The BN-based algorithms described in this study enable modeling of infrastructure systems at a level of detail and complexity not previously possible. Implementation of these methodologies is a step towards making more efficient and more effective engineering decisions about our infrastructure systems both now and for the future.

References

Akers, S. B., "Binary decision diagrams," *IEEE Transactions of Computers*, Vol. C-27, No. 6, pp. 509-516, June 1978.

Au, S. K., Zhang, F. L., and Ni, Y. C., "Bayesian operational modal analysis: Theory, computation, practice," *Computers and Structures*, Vol. 126, pp. 3-14, September 2013.

Beck, J. L., and Au, S. K., "Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation," *Journal of Engineering Mechanics*, Vol. 128, No. 4, pp. 380-391, 2002.

Benaddy, M., and Wakrim, M., "Cutset enumerating and network reliability computing by a new recursive algorithm and inclusion exclusion principle," *International Journal of Computer Applications*, Vol. 45, No. 16, pp. 22-25, May 2012.

Bensi, M. T., Der Kiureghian, A., and Straub, D., "A Bayesian Network Methodology for Infrastructure Seismic Risk Assessment and Decision Support," *Pacific Earthquake Engineering Research Center (PEER) Report No. 2011/02*, University of California, Berkeley, March 2011.

Bensi, M., Der Kiureghian, A., and Straub, D., "Efficient Bayesian network modeling of systems," *Reliability Engineering and System Safety*, Vol. 112, pp. 200-213, 2013.

Bernal, D., "New approaches to analysis and testing of mechanical and structural systems: state space identification," *Workshop at the International Centre for Mechanical Sciences*, June 18-22, 2007.

Birolini, A., *Reliability Engineering: Theory and Practice*, 4th ed., Berlin: Springer, 2004.

Bobbio, A., Portinale, L., Minichino, M., and Ciancamerla, E., "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," *Reliability Engineering and System Safety*, Vol. 71, No. 3, pp. 249-260, 2001.

- Boudali, H., and Dugan, J. B., "A discrete-time Bayesian network reliability modeling and analysis framework," *Reliability Engineering and System Safety*, Vol. 87, pp. 337-349, 2005.
- Bryant, R. E., "Graph-based algorithms for Boolean function manipulation," *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 677-691, August 1986.
- Cheok, M. C., Parry, G. W., and Sherry, R. R., "Use of importance measures in risk-informed regulatory applications," *Reliability Engineering and System Safety*, Vol. 60, No. 3, pp. 213-226, June 1998.
- Dean, T., and Kanazawa, K., "A model for reasoning about persistence and causation," *Computational Intelligence*, Vol. 5, No. 3, pp. 142-150, 1989.
- Dechter, R., "Bucket Elimination: a Unifying Framework for Reasoning," *Artificial Intelligence*, Vol. 113, pp. 41-85, 1999.
- Der Kiureghian, A., and Song, J., "Multi-scale reliability analysis and updating of complex systems by use of linear programming," *Reliability Engineering and System Safety*, Vol. 93, pp. 288-297, 2008.
- Der Kiureghian, A., and Wung, C. D., "A computer-assisted learning system for random vibrations," *Computers and Structures*, Vol. 43, No. 5, pp. 975-993, 1992.
- Fard, N. S., "Determination of minimal cut sets of a complex fault tree," *Computers and Industrial Engineering*, Vol. 33, No. 1-2, pp. 59-62, October 1997.
- Gasparini, D. A. et al., "Random Vibration of Cascaded Secondary System," *Proceedings of the 7th SMIRT Conference*, Chicago, Illinois, p. 445-451, 1983.
- Ghanem, R., and Ferro, G., "Health monitoring for strongly non-linear systems using the ensemble Kalman filter," *Structural Control and Health Monitoring*, Vol. 13, No. 1, pp. 245-259, January/February 2006.
- Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., *Markov chain Monte Carlo in practice*, 1st ed., Vol. 17, London: Chapman & Hall, pp. 486, 1996.
- Hauck, E. L., "Data compression using run length encoding and statistical encoding," U.S. Patent 4 626 829 A, December 2, 1986.
- Jensen, F. V., and Nielsen, T. D., *Bayesian Networks and Decision Graphs*, 2nd ed., New York: Springer, 2007.
- Johansson, R., Verhaegen, M., and Chou, C. T., "Stochastic Theory of Continuous-Time State-Space Identification," *IEEE Transactions on Signal Processing*, Vol. 47, No. 1, pp. 41-51, January 1999.

Katafygiotis, L. S., and Yuen, K. V., “Bayesian spectral density approach for modal updating using ambient data,” *Earthquake Engineering and Structural Dynamics*, Vol. 30, No. 8, pp. 1103-1123, August 2001.

Kim, M. C., “Reliability Block Diagram with General Gates and its Application to System Reliability Analysis,” *Annals of Nuclear Energy*, Vol. 38, pp. 2456-2461, 2011.

Kottapalli, V. A., Kiremidjian, A. S., Lynch, J. P., Carryer, E., Kenny, T. W., Law, K. H., and Lei, Y., “Two-tiered wireless sensor network architecture for structural health monitoring,” *Proceedings of SPIE*, Vol. 5057, 2003.

Lauritzen, S. L., and Jensen, F., “Stable local computation with conditional Gaussian distributions,” *Statistics and Computing*, Vol. 11, No. 2, pp. 191-203, April 2001.

Li, J., Qian, Y., and Liu, W., “Minimal cut-based recursive decomposition algorithm for seismic reliability evaluation of lifeline networks,” *Earthquake Engineering and Engineering Vibration*, Vol. 6, No. 1, pp. 21-28, March 2007.

Lutes, L. and Sarkani, S., *Random Vibrations: Analysis of Structural and Mechanical Systems*, Burlington, MA: Elsevier, 2004.

Madsen, A. L., “Belief update in CLG Bayesian networks with lazy propagation,” *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2012.

Mahadevan, S., Zhang R., and Smith, N., “Bayesian networks for system reliability reassessment,” *Structural Safety*, Vol. 23, pp. 231-251, 2001.

Modarres, M., Kaminskiy, M., and Krivtsov, V., *Reliability Engineering and Risk Analysis: A Practical Guide*, 2nd ed., Boca Raton, Florida: CRC Press, 2010.

Murphy, K. P., “The Bayes Net Toolbox for Matlab,” *Computing Science and Statistics: Proceedings of the Interface*, Vol. 33, October 2001.

Murphy, K. P., “Dynamic Bayesian networks: representation, inference and learning,” *Doctoral Thesis*, 2002.

Neal, R. M., “Probabilistic inference using Markov chain Monte Carlo methods,” *Technical Report CRG-TR-93-1*, University of Toronto: Department of Computer Science, September 1993.

Nielsen, T. D., Willemin, P. H., and Jensen, F. V., “Using ROBDDs for inference in Bayesian networks with troubleshooting as an example,” *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, Stanford University, Stanford, CA, pp. 426-435, June 30 – July 3, 2000.

Oppenheim, A. V., Schafer, R. W., and Buck, J. R., *Discrete-Time Signal Processing*, 2nd ed., Englewood Cliffs, New Jersey: Prentice-Hall, 1999.

Ostrom D., "Database of seismic parameters of equipment in substations," *Report to Pacific Earthquake Engineering Research Center*, <http://peer.berkeley.edu>, 2004.

Pages, A., and Gondran, M., *System Reliability: Evaluation and Prediction in Engineering*, New York: Springer-Verlag, 1986.

Rausand, M., and Hoyland, A., *System Reliability Theory: Models, Statistical Methods, and Applications*, 2nd ed., Hoboken, New Jersey: John Wiley & Sons, 2004.

Rezaeian, S., and Der Kiureghian, A., "Stochastic Modeling and Simulation of Ground Motions for Performance-Based Earthquake Engineering," *Pacific Earthquake Engineering Research Center (PEER) Report No. 2010/02*, University of California, Berkeley, June 2010.

Rice, S. O., "Mathematical analysis of random noise," *Bell System Technical Journal*, Vol. 23, No. 3, pp. 282-332, July 1944.

Salmeron, A., Cano, A., and Moral, S., "Importance sampling in Bayesian networks using probability trees," *Computational Statistics and Data Analysis*, Vol. 34, pp. 387-413, 2000.

Shin, Y. Y., and Koh, J. S., "An algorithm for generating minimal cutsets of undirected graphs," *Korean Journal of Computational and Applied Mathematics*, Vol. 5, No. 3, pp. 681-693, September 1998.

Sohn, H., and Law, K. H., "A Bayesian probabilistic approach for structure damage detection," *Earthquake Engineering and Structural Dynamics*, Vol. 26, No. 12, pp. 1259-1281, December 1997.

Song, J., and Der Kiureghian, A., "Importance measures for systems with dependent components," *Proceedings of the 9th International Conference on Structural Safety and Reliability*, Rome, Italy, pp. 1431-1438, June 19 – 23, 2005.

Song, J., and Ok, S. Y., "Multi-scale system reliability analysis of lifeline networks under earthquake hazards," *Earthquake Engineering and Structural Dynamics*, Vol. 39, pp. 259-279, 2010.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., and Cowell, R. G., "Bayesian Analysis in Expert Systems," *Statistical Science*, Vol. 8, No. 3, pp. 219-247, 1993.

Straser, E. G., and Kiremidjian, A. S., "Monitoring and evaluating civil structures using measured vibration," *Proceedings of SPIE*, Vol. 2719, 1996.

Straub, D., "Stochastic modeling of deterioration processes through dynamic Bayesian networks," *Journal of Engineering Mechanics*, Vol. 135, No. 10, pp. 1089-1099, October 2009.

Suh, H., and Chang, C. K., "Algorithms for the minimal cutsets enumeration of networks by graph search and branch addition," *Proceedings of the 25th Annual IEEE Conference on Local Computer Networks*, Tampa, FL, pp. 100-107, November 8 – 10, 2000.

Tan, Z., "Minimal cut sets of s - t networks with k -out-of- n nodes," *Reliability Engineering and System Safety*, Vol. 82, No. 1, pp. 49-54, October 2003.

Torres-Toledano, J. G., and Succar, L. E., "Bayesian networks for reliability analysis of complex systems," *Lecture Notes in Artificial Intelligence 1484*, pp. 195-206, 1998.

Vanik, M. W., and Beck, J. L., "A Bayesian probabilistic approach to structural health monitoring," *Proceedings of the International Workshop on Structural Health Monitoring*, Lancaster, PA, pp. 140-151, 1997.

Vanik, M. W., Beck, J. L., and Au, S. K., "Bayesian probabilistic approach to structural health monitoring," *Journal of Engineering Mechanics*, Vol. 126, No. 7, pp. 738-745, July 2000.

Vesely, W. E., Davis, T. C., Denning, R. S., and Saltos, N., "Measures of Risk Importance and Their Applications (NUREG/CR-3385)," Division of Risk Analysis, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, July 1983.

Vesely, W. E., Goldberg, F. F., Roberts, N. H., and Haasl, D. F., "Fault Tree Handbook (NUREG-0492)," Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission, January 1981.

Vinod, G., Kushwaha, H. S., Verma, A. K., and Srividya, A., "Importance measures in ranking piping components for risk informed in-service inspection," *Reliability Engineering and System Safety*, Vol. 80, No. 2, pp. 107-113, May 2003.

Weber, P., and Jouffe, L., "Complex system reliability modeling with Dynamic Object Oriented Bayesian Networks (DOOBN)," *Reliability Engineering and System Safety*, Vol. 91, pp. 149-162, 2006.

Welch, G., and Bishop, G., "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Technical Report 95-041, 1995.

Wu, S., and Beck, J. L., "Synergistic combination of systems for structural health monitoring and earthquake early warning for structural health prognosis and diagnosis," *Proceedings of SPIE*, Vol. 8348, 2012.

Yang, J. N., Lin, S., Huang, H., and Zhou, L., “An adaptive extended Kalman filter for structural damage identification,” *Structural Control and Health Monitoring*, Vol. 13, No. 4, pp. 849-867, July/August 2006.

Yang, J. N., Pan, S., and Huang, H., “An adaptive extended Kalman filter for structural damage identifications II: unknown inputs,” *Structural Control and Health Monitoring*, Vol. 14, No. 3, pp. 497-521, April 2007.

Yeh, W.-C., “A new algorithm for generating minimal cut sets in k -out-of- n networks,” *Reliability Engineering and System Safety*, Vol. 91, No. 1, pp. 36-43, January 2006.

Yuan, C., and Druzdzel, M. J., “Importance sampling algorithms for Bayesian networks: Principles and performance,” *Mathematical and Computer Modelling*, Vol. 43, No. 9-10, pp. 1189- 1207, 2006.

Yuen, K., *Bayesian Methods for Structural Dynamics and Civil Engineering*, Singapore: Wiley, 2010.

Yuen, K. V., and Katafygiotis, L. S., “Bayesian modal updating using complete input and incomplete response noisy measurements,” *Journal of Engineering Mechanics*, Vol. 128, No. 3, pp. 340-350, March 2002.

Ziv, J., and Lempel, A., “A Universal Algorithm for Sequential Data Compression,” *IEEE Transactions on Information Theory*, Vol. 23, No. 3, pp. 337-343, May 1977.