

Compression algorithm for Bayesian network modeling of binary systems

I. Tien & A. Der Kiureghian

University of California, Berkeley, USA

ABSTRACT: A Bayesian Network (BN) is a useful tool for analyzing the reliability of systems. The BN framework is limited, however, by the size and complexity of the system that can be tractably modeled. Each node in a BN graph is associated with a Conditional Probability Table (CPT), the size of which grows exponentially with the number of connected nodes in the graph, presenting a memory storage challenge in constructing and analyzing the BN. In this paper, we look at binary systems, where components of the system are in either one of two states, survival or failure, and the component states deterministically define the system state. This analysis is particularly useful for studying the reliability of infrastructure systems, where, e.g., the states of individual gas pipelines or roads directly impact the state of the overall natural gas or transportation system. We present a compression algorithm for the CPTs of such systems so that they may be modeled on a larger scale as BNs. We apply our algorithm to an example system and evaluate its performance compared to an existing algorithm.

1 INTRODUCTION

A Bayesian network (BN) is a probabilistic framework well suited for modeling systems and analyzing their reliability. A BN is a directed acyclic graph comprised of nodes and links. In the BN system model, the components of the system are represented as the nodes of the graph, and the directed links between the nodes represent the dependencies between the components. Since the state of the system depends on the states of its components, the node representing the system is defined as a child of the nodes representing the components, i.e., links are directed from component nodes to the system node. A BN model enables analysis of the contributions of individual components to the overall system reliability and the identification of critical components within the system.

As information in the model is handled probabilistically, the BN framework is particularly useful for decision-making under uncertainty for infrastructure systems, where component demands and capacities are uncertain. In addition, the BN framework allows for updating of the network as new information, or evidence, becomes available. When evidence on one or more variables is entered into the BN, the information propagates through the network to yield updated probabilities of the system performance in light of the new information. This enables decision making about the system based on the most up-to-date information.

The BN framework is limited, however, by the size and complexity of the system that can be tractably modeled. Each node in the BN is associated with a conditional probability table (CPT), which defines the conditional probability mass function of the node, given each mutually exclusive combination of the

states of its parent nodes. Consequently, the size of the CPT grows exponentially with the number of parents of the node. For example, if every node has m states, then the CPT has m^{n+1} entries, where n denotes the number of parents. Thus, in a binary system with 100 binary components (i.e., each component having two states, e.g., survival or failure), the CPT for the system node consists of $2^{101} = 2.5E30$ individual terms. This poses a significant memory storage challenge in constructing and analyzing the BN. For this reason, BN modeling of systems has been limited to small systems, typically of no more than 20 two-state components.

In this paper, we present a novel compression algorithm to enable the modeling of large, complex systems as BNs. The algorithm encodes the system CPT in compressed form, both in the initial construction of the BN and in the subsequent calculations for inference, to achieve significant savings in memory storage. The algorithm, however, necessitates longer computation time. Thus, a trade-off is made between the demand for memory storage, which has a hard limit, and the demand for computation time. Several features are added to the compression algorithm to reduce the computation time. An example application demonstrates the memory and computational demands of the proposed algorithm versus the most widely used existing algorithm.

2 RELATED WORK

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,

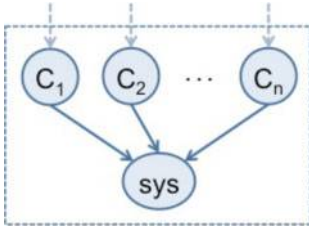


Figure 1. BN of a system with n components.

such as reliability block diagrams (Torres-Toledano and Succar 1998) and fault trees (Bobbio et al. 2001). However, these studies have been limited to small systems comprised of no more than ten components.

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. It is this assumption that bounds the size of the ROBDD and enables modeling of larger systems. For more general systems, this single-fault assumption cannot be guaranteed, and the gains from using the ROBDD may not be applicable. Finally, a topology optimization algorithm is proposed in Bensi et al. (2013) to develop a chain-like BN model of the system with minimal clique sizes. The optimization program, however, must consider the permutation of all component indices and, therefore, may itself become intractably large for large systems.

3 THE PROPOSED METHOD

3.1 Binary systems

Consider a system consisting of binary components, i.e., where each component is in one of two possible states, survival or failure. We assume the component states deterministically define the system state and that the system is also binary, i.e., for any combination of the component states, the system is in either a survival or a failure state. This 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. The BN of a system with n components C_1, \dots, C_n is shown in Figure 1, where sys denotes the system node.

Table 1. Example CPT for a binary system with n components.

C_1	...	C_{n-1}	C_n	sys
0	...	0	0	0
0	...	0	1	0
0	...	1	0	0
0	...	1	1	1
...
1	...	0	0	0
1	...	0	1	1
1	...	1	0	0
1	...	1	1	1

As indicated by the dotted arrows in Figure 1, the BN model accommodates parent nodes to the component nodes, which may represent demands on the components resulting from a hazard. Here, the compression algorithm developed focuses on the system description part of the BN enclosed in the dashed box, which models the system performance. In addition, while the algorithm is developed for binary systems, it 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.

When the component states deterministically define the system state, the CPT associated with the system node has a special property. 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. Table 1 shows an example of such a CPT for a binary system with binary components. As seen in the rightmost column of the table, the vector representing the system state given each distinct combination of component states is comprised solely of 0s and 1s and has the size $2^n \times 1$. The proposed compression algorithm takes advantage of this property.

3.2 Compression algorithm

The developed algorithm utilizes compression techniques, including run-length encoding and the classical Lempel-Ziv algorithm. A run is defined as consecutive bits of the same value. Run-length encoding stores runs in a dataset as a data value and count, making it well suited for data with many repeated values. However, mixed values are stored literally, which results in little gain for mixed data. Algorithms 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. The proposed compression algorithm uses both these ideas to compress the system CPT of the BN.

A binary system can be defined in terms of its minimum-cut sets (MCSs) or minimum-link sets (MLSs). A MCS is a minimum set of components whose joint failure constitutes failure of the system; if any one MCS fails, the system fails. Similarly, a MLS is a minimum set of components whose joint survival constitutes survival of the system; if any one MLS survives, the system survives. Each combination of component states in the system CPT (see Table 1) is checked against MCSs or MLSs to determine the system state. For a given combination of component states, this is the rightmost value in the row, the last column in Table 1. This value is then encoded in a compressed form, with repeated values in the column encoded as runs, and patterns in the column encoded as phrases with an accompanying dictionary of phrases. Note that we only need to compress the vector of system states; there is no need to compress the entire CPT. This is because the component states in any row can be determined in terms of the row number due to the specific pattern used in defining the component states in the table. Specifically, the state of component i , s_i , in row m of the CPT for a system of n total components is determined according to the rule

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

where $\text{ceil}(x)$ is the value of x rounded up to the nearest integer. The result from employing the above algorithm is that the full system CPT becomes encoded as a compressed combination of runs and phrases. Typically, the size of this data is orders of magnitude smaller than the size of the CPT.

3.3 Inference

Once the system CPT for the BN has been constructed in a compressed form, inference is required to draw conclusions about the system. There are both exact and approximate methods for inference. Approximate methods are generally sampling based. However, for a reliable system where failure events are rare, these methods often yield poor convergence due to the unlikelihood of the events that are being modeled, and the number of simulations required to achieve a sufficiently large sample is prohibitive. Therefore, exact methods of inference are preferred and are considered here.

There are two major algorithms used for exact inference: the junction tree (JT) algorithm (Spiegelhalter et al. 1993) and the variable elimination (VE) algorithm (Dechter 1999). The advantage of the JT algorithm comes from breaking a network down into smaller structures, or cliques. However, for the network we have in Figure 1, the BN 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. For this reason, here the VE algorithm is used for inference.

As its name suggests, the VE algorithm works by eliminating the nodes in the network, one by one, to arrive at the query node, the node for which the posterior (updated) probability distribution is of interest. 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 that is used during the next step of elimination. For the system given in Figure 1, suppose we are interested in the posterior distribution of the state of component 1 given a particular state of the system. The VE calculation for this query is

$$\begin{aligned} & P(C_1 | \text{sys}) \\ &= P(C_1) \sum_{C_2} P(C_2) \dots \sum_{C_{n-1}} P(C_{n-1}) \sum_{C_n} P(C_n) CPT_{\text{sys}} \\ &= P(C_1) \sum_{C_2} P(C_2) \dots \sum_{C_{n-1}} P(C_{n-1}) \lambda_n \\ &= \dots \\ &= P(C_1) \lambda_2 \end{aligned} \quad (2)$$

where CPT_{sys} is the compressed system CPT and λ_i is the intermediate factor created after elimination of node (component) i . Once the initial CPT_{sys} has been compressed, the key to successfully performing inference for a large system is that each subsequent λ_i must also be stored using the same compression algorithm as used for CPT_{sys} . The algorithm for inference that we have developed employs the VE method and is able to handle the compressed CPT_{sys} and λ_i without decompressing or recompressing. In this way, the memory requirements for storing both the initial system CPT and the intermediate factors during inference are significantly reduced, enabling BN modeling of large systems.

3.4 Example system

We start with the 8-component example system shown in Figure 2(a), which is adopted from Bensi et al. (2013). The system consists of a series subsystem $\{C_4, C_5, C_6\}$, and a parallel subsystem $\{C_1, C_2, C_3\}$. Because the objective is to see how the proposed algorithm scales with increasing system size, we increase the number of components in these subsystems and analyze the performance of the algorithm as the number of components increases. We note that, as pointed out by Der Kiureghian and Song (2008), Song and Ok (2010) and Bensi et al. (2013), the system in Figure 2(a) can be more efficiently represented as a system of three super-components, each super-component representing a series or parallel subsystem. However, here we disregard this advantage in order to investigate the system size effect.

We increase the number of components in the series subsystem so that the total number of components in the system is n , as in Figure 2(b). We also increase

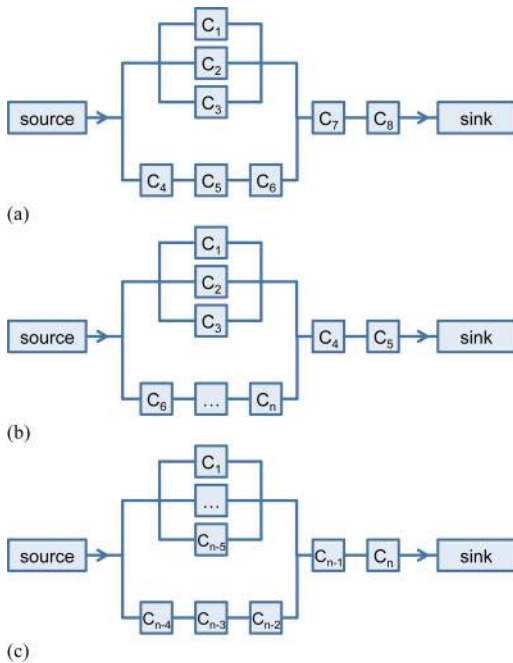


Figure 2. Example systems: (a) basic case, (b) with increased components in series subsystem, and (c) with increased components in parallel subsystem.

the number of components in the parallel subsystem so that the total number of components in the system is n , as in Figure 2(c). The resulting analysis of these two expanded systems, Figure 2(b) and Figure 2(c), with variable n shows how the proposed compression algorithm performs compared to existing algorithms to analyze systems of increasing size.

In order to determine the state of the system for each distinct combination of the states of the components, we use the MCS representation of the system. A similar MLS formulation of the system can also be used. The MCSs of the system in Figure 2(a) are $\{(C_1, C_2, C_3, C_4), (C_1, C_2, C_3, C_5), (C_1, C_2, C_3, C_6), (C_7), (C_8)\}$. The MCSs of the system in Figure 2(b) with the expanded 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 in Figure 2(c) with the expanded parallel subsystem are $\{(C_1, \dots, C_{n-5}, C_{n-4}), (C_1, \dots, C_{n-5}, C_{n-3}), (C_1, \dots, C_{n-5}, C_{n-2}), (C_{n-1}), (C_n)\}$.

4 RESULTS

4.1 Inference

The BN is initialized with a prior probability of failure for the parallel components, C_1, C_2, C_3 in Figure 2(b), and C_1, \dots, C_{n-5} , in Figure 2(c), of 0.2, and a prior probability of failure for the components in series, C_4, \dots, C_n , in Figure 2(b), and C_{n-4}, \dots, C_n , in Figure 2(c), of 0.01. We are interested in updating the probabilities

of failure of the system and components given new information, or evidence. The proposed new algorithm consists of first implementing the developed compression algorithm for the full system CPT to construct the BN, and then implementing the developed inference algorithm to perform VE on the compressed matrices.

For illustration purposes, in the following we conduct inference using component 1. Figure 3(a) shows the updated probabilities of system failure given the evidence that component 1 has failed, i.e., $C_1 = 0$. Figure 3(b) shows the updated probabilities of failure of component 1, given the evidence that the system has failed. 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 . The sequence connected by a solid line indicates the results from increasing the number of components in the series subsystem (Figure 2b), and the sequence connected by the dashed line indicates the results from increasing the number of components in the parallel subsystem (Figure 2c). The results in Figure 3 show that the proposed algorithm successfully performs both forward and backward inference. In Figure 3(a), we see that the 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 minimum cut sets $\{C_{n-1}\}$ and $\{C_n\}$.

In Figure 3(b), 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 .

4.2 Memory storage

To analyze the performance of the new algorithm compared to existing algorithms, we first look at memory storage. Figure 4 shows the maximum number of values that must be stored in memory during the running

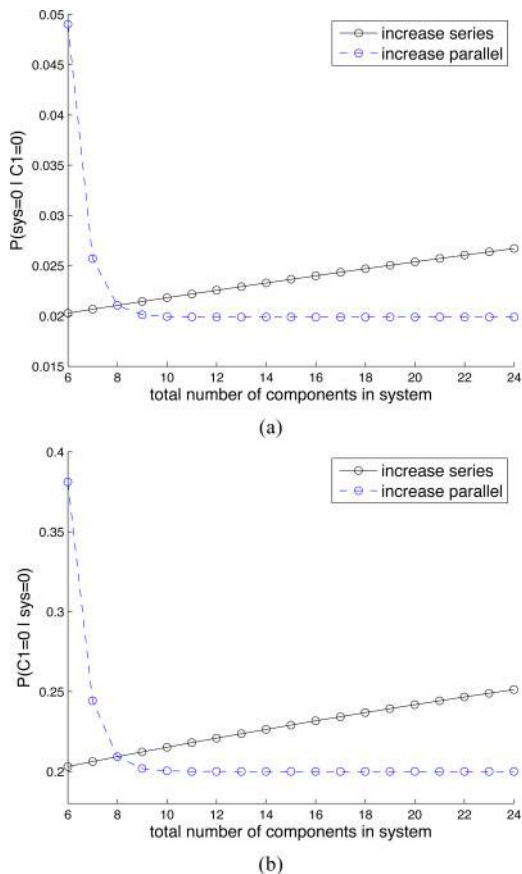


Figure 3. Updated probabilities of (a) system state given component state, and (b) component state given system state with increasing system size BN of a system with n components.

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. The circle marks indicate the maximum number of values stored when running the new algorithm, including the values both in the compressed CPT and in the accompanying phrase dictionary. The squares indicate the maximum number of elements stored during the construction of the BN using the existing JT algorithm, as implemented in the Bayes Net Toolbox by Murphy (2001). 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 2b), and from increasing the number of components in the parallel subsystem (Figure 2c).

Figure 4 shows that the proposed new algorithm achieves significant gains in memory storage demand compared to the existing algorithm. For the existing

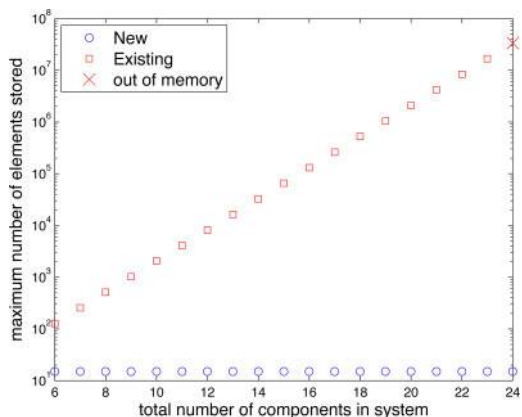


Figure 4. Maximum number of elements that must be stored using the new vs. existing algorithm as a function of system size.

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 on our 32Gb RAM computer for a system comprised of more than 24 components. The memory storage demand using the proposed new algorithm, however, remains constant, even as the number of components in the system increases.

4.3 Computation time

Figure 5 shows the computation times required to run the algorithms with increasing system size. In Figure 5(a), 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 solid bars indicate the time required to perform inference on the system given $\{C_1 = 0\}$ and the time to perform 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 time required to initialize the BN using the existing JT algorithm. The next two bars with diagonal hatching indicate the time required to perform inference on the system given information on the component C_1 , and to perform inference on the component C_1 given information on the system, respectively, 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 in Figure 5(a) are for the case where the size of the system increases by increasing the number of components in the series subsystem. Figure 5(b) compares the computation times for the proposed algorithm for the two systems in Figures 3(a) and 3(b). Again the computation times are broken down into the component for compressing the system CPT, to perform inference on the system given the state of

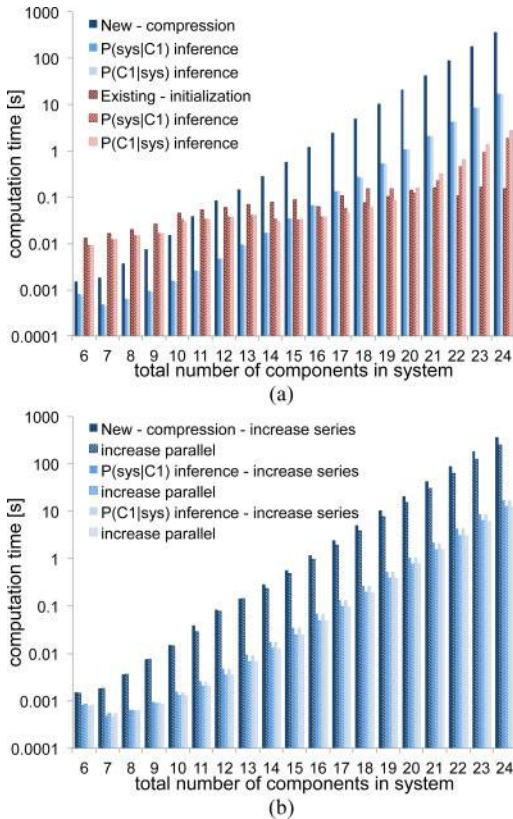


Figure 5. Computation times as function of system size: (a) proposed algorithm vs. existing JT algorithm, (b) proposed algorithm when increasing the number of components in the series vs. parallel subsystems.

component C_1 , and to perform 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.

Taking Figures 4 and 5 together, we see the classic storage space-computation time trade-off as described in Dechter (1999). In Figure 5(a), we see that the new algorithm requires longer computation times compared to the existing JT algorithm. However, the time to perform inference for both the new and existing algorithms is exponentially increasing with the system size. 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. In contrast, computation time is more flexible. Indeed, various recourses are available to address the computational time, such as parallel computing. While

reliability analyses will take longer using the new algorithm, some problems simply cannot be solved using existing algorithms.

In Figure 5(b), by comparing the results of increasing the size of the series subsystem vs. the parallel subsystem, 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. 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.

5 CONCLUSION

We have developed a compression algorithm that significantly reduces the memory storage requirements during the construction of a BN for system reliability analysis. In addition, we have developed a variable elimination (VE) algorithm to perform inference using compressed matrices. It is shown that by implementing the proposed algorithm, the memory storage demand during construction of the BN and during inference not only does not exponentially increase as the number of components in the system increases, but essentially remains constant. Together, these algorithms enable large systems to be modeled as BNs for system reliability analysis.

ACKNOWLEDGEMENTS

The first author gratefully acknowledges support from the National Science Foundation Graduate Research Fellowship. Additional support is provided by the National Science Foundation Grant No. CMMI-1130061, which is also gratefully acknowledged.

REFERENCES

- 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., A. Der Kiureghian and D. Straub (2013). Efficient Bayesian network modeling of systems. *Reliability Engineering and System Safety*, 112:200–213.
- 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.

- 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.
- Murphy, K. P., "The Bayes Net Toolbox for Matlab," *Computing Science and Statistics: Proceedings of the Interface*, Vol. 33, October 2001.
- Nielsen, T. D., Wuillemin, 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.
- 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.
- 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.
- 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.