Algorithms for Bayesian Network Modeling, Inference, and Reliability Assessment for Multistate Flow Networks

Yanjie Tong[1] and Iris Tien, Ph.D., A.M.ASCE[2]

Abstract: The Bayesian network (BN) is a useful tool for the modeling and reliability assessment of civil infrastructure systems. For a system comprising many interconnected components, it captures the probabilistic dependencies between components and system performance, with inference in the BN informing decision making in the management of these systems. However, one of the major challenges in the BN modeling of infrastructure systems is the exponentially increasing computational complexity as the number of components in the system increases. Previously, algorithms have been developed for BN modeling of binary systems. Compared with binary systems, multistate system modeling provides a more detailed description of system reliability and enables the analysis of flow instead of connectivity networks. However, the dimensionality of the problem also increases. This paper advances the state of the art in BN modeling of complex networks by presenting new algorithms for constructing the BN model for multistate components and systems and performing exact inference over these models. The results support reliability assessment of civil infrastructure flow systems. Specifically, the authors present a new lossless compression algorithm for initial construction of the BN model and simultaneous preprocessing of intermediate factors for inference. These significantly reduce memory storage requirements for the BN. Two heuristics are described to further increase computational efficiency. The new algorithms are applied to an example infrastructure system. The ability to conduct inference across the network is demonstrated and performance measured compared to existing algorithms in terms of both memory storage and computation time. The proposed algorithms are shown to achieve exponentially increasing data compression with a stable increased computation time ratio, enabling larger multistate flow networks to be modeled as BNs than previously possible. **DOI:** 10.1061/(ASCE)CP.1943-5487.0000699. *© 2017 American Society of Civil Engineers.*

Author keywords: Civil infrastructure systems; Bayesian networks; Multistate system modeling; Inference; Algorithms; Reliability assessment.

Introduction

Infrastructure systems are critical to everyday lives and to the health, safety, and functioning of society. They enable researchers to access people's daily needs, including water, electricity, and food. However, these systems are aging and subject to hazards of increasing frequency and severity. In this environment, it is important that infrastructure systems continue to function reliably. Rigorous reliability assessments support decision making in infrastructure management to achieve this.

Any given civil infrastructure system comprises many interconnected components. The functioning of individual components results in the provision of services by the systems overall. Assessing the impact of component performance on system performance enables a stakeholder to identify the most critical components, and prioritize decisions for inspection, repair, or replacement of these system elements. The objective is to create more reliable systems under both normal operating and hazardous conditions, leading to improved community outcomes (Johansen et al. 2016).

The Bayesian network (BN) is a useful framework under which to perform infrastructure reliability assessments. Given the uncertainties associated with component performance and the hazards components are subjected to, the BN models component states as random variables and captures the probabilistic dependencies between component and system performance. In addition, in an environment of evolving information, for example, where inspections offer new insights into the current states of components, any information entered into the BN propagates through the network to update assessments of the systems. Finally, the BN as a graphical framework enables transparent modeling of systems to facilitate adoption by end-users.

One of the major challenges in the BN modeling of infrastructure systems is the exponentially increasing computational complexity as the number of components in the system increases. Previously, algorithms were proposed for BN modeling of binary systems (Tien and Der Kiureghian 2013) to enable the study of larger infrastructure systems within the BN framework (Tien and Der Kiureghian 2016). However, this binary system formulation, where components and the system are in one of two states (e.g., failure of survival) is not sufficient to describe the status of many infrastructure components and systems (Tong and Tien 2016). This is true for infrastructures including water, gas line, and transportation networks, which are characterized by flow (e.g., of water supply, natural gas, and vehicles) across the network. In these cases, if a component is functioning at 50% of its maximum capacity, this cannot be sufficiently defined as failure or survival. Compared with binary systems, multistate system modeling provides a more

[1]Ph.D. Student, School of Civil and Environmental Engineering, Georgia Institute of Technology, 631 Cherry St., Atlanta, GA 30332-0355 (corresponding author). ORCID: https://orcid.org/0000-0001-5574-3889 E-mail: yjtong@gatech.edu

[2]Assistant Professor, School of Civil and Environmental Engineering, Georgia Institute of Technology, 631 Cherry St., Atlanta, GA 30332-0355. E-mail: itien@ce.gatech.edu

Note. This manuscript was submitted on June 8, 2016; approved on March 22, 2017; published online on June 19, 2017. Discussion period open until November 19, 2017; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Computing in Civil Engineering***, © ASCE, ISSN 0887-3801.

detailed description of system reliability and enables the analysis of flow instead of connectivity networks. However, the dimensionality of the problem also increases. This paper describes new algorithms developed for BN modeling, inference, and reliability assessment for these multistate flow networks.

The rest of the paper is organized as follows. The authors first provide a brief background on the use of BNs for system reliability assessment and describe the proposed BN system formulation. The authors then present new algorithms for BN modeling of multistate flow networks and for performing exact inference over these models. These include the proposed approach using compression to reduce the memory storage requirements of the conditional probability tables associated with the BN and two heuristics to increase the computational efficiency of the method. The authors apply the algorithms to an example infrastructure system to demonstrate their use for reliability assessment. Finally, the authors assess the performance of the proposed algorithms compared to existing methods in terms of both memory storage and computation time.

Background and Related Work

Bayesian Networks for System Reliability Assessment

Previous studies on the use of BNs for modeling system performance have focused on generating BNs from conventional system modeling methods, such as reliability block diagrams (Torres-Toledano and Succar 1998; Kim 2011) and fault trees (Bobbio et al. 2001). These and other studies using BNs for system reliability assessment (Mahadevan et al. 2001; Boudali and Dugan 2005; Khakzad et al. 2011; Tien and Der Kiureghian 2015) have all assumed binary component and system states. This allows the modeling of systems characterized by connectivity, such as the power distribution network in Tien and Der Kiureghian (2017), but not flows.

In the study of multistate systems, Bouissou and Pourret (2003) propose a BN-based method for performance evaluation of systems with multiple states. However, the focus is on troubleshooting, or identification of single causes of system failure. The assumption of single-fault failures does not hold in the assessment of civil infrastructure systems. The reduced capacity of multiple components may lead to reduction in system performance. Gu and Yang (2013)

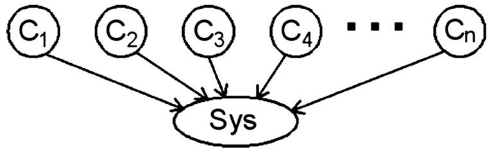

Fig. 1. Bayesian network model of a system comprising *n* components

use BNs to assess the reliability of multistate systems. However, the system studied consists of only six components, and even with this small number of components, intermediate nodes are introduced in the BN to enable computationally tractable calculations using the traditional BN modeling method.

Bensi et al. (2013) propose a method for more efficient modeling of BNs, including for multistate systems. The study takes a topology optimization approach to address the system size limitation of BN models. However, the optimization algorithm must consider all permutations of the indices of system components, and therefore may itself become intractably large for large infrastructure systems. Finally, a method based on the flow conservation law is proposed in Yeh (2013) to assess the reliability of a multistate flow network. However, the focus is on modeling deterioration effects, rather than on quantifying the importance of individual component performance on overall system reliability. In addition, the systems considered in this paper do not necessarily obey the flow conservation law, i.e., that the flow into any node is equal to the flow out of that node. For example, in the tested systems, a given level of flow may enter a system component, but given the state of that component, the flow out may be different. Thus, new methods are required for the BN modeling of multistate networks.

BN System Formulation

For a network of n components, the BN model is as shown in Fig. 1. The state of the system, represented as a system node, sys, is dependent on the states of each of its constituent components, represented as component nodes $C_1, \ldots, C_n$. In BN terminology, $C_1, \ldots, C_n$ are called parents of sys.

The BN is a probabilistic graphical model. For the BN, each node must be associated with a conditional probability table (CPT), which gives the probability distribution of that node given each of the mutually exclusive combinations of states of its parents. Nodes that do not depend on other nodes are defined by marginal probability distributions. The reader is referred to texts such as Jensen and Nielsen (2007) for further details on BNs.

Now suppose that the components and system can be in one of multiple possible states, e.g., States 0, 1, 2, 3, or 4 denoting discretized values of flow capacity 0, 25, 50, 75, and 100% of maximum capacity, respectively. An example of the CPT associated with the system node for this multistate flow network is shown in Table 1. Let n denote the number of components and m the number of states of each component. For the columns indicating system states, the authors use m columns to represent whether the system is in that specific state. If so, the value in the column is 1; otherwise, it is 0.

As the number of components in the system increases, the size of the CPT as shown in Table 1 increases exponentially. In general, the system states in the CPT are represented by $m \times m^n$ elements, as shown in the m right-hand columns in Table 1. Because of

Table 1. Example Conditional Probability Table for a Multistate System

Row number	C_1	$\cdots$	C_{n-1}	C_n	$sys = 0$	$sys = 1$	$sys = 2$	$\ldots$	$sys = m-1$
1	0	$\cdots$	0	0	1	0	0	$\ldots$	0
2	0	$\cdots$	0	1	1	0	0	$\ldots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	—	—	—	$\vdots$	—
$m^n - 4$	4	$\cdots$	4	0	0	1	0	$\cdots$	0
$m^n - 3$	4	$\cdots$	4	1	0	0	1	$\cdots$	0
$m^n - 2$	4	$\cdots m^n - 1 \cdots$	4	2	0	0	0	$\cdots$	1
$m^n - 1$	4	$\cdots$	4	3	0	0	0	$\cdots$	0
m^n	4	$\cdots$	4	4	0	0	0	$\cdots$	0

mutually exclusive system states, knowing the column in which the 1 value appears allows one to infer the values in the other columns for that row. This reduces the number of elements to m^n. However, this value is still exponentially increasing with the number of components in the system being modeled. For example, for a five-state system of $n = 100$ components, the full representation of the CPT includes a minimum of $5^{100} = 7.9 \times 10^{69}$ elements. The exponential increase in the size of the CPT poses a significant memory storage challenge in constructing and analyzing the BN. It quickly renders the model intractable, necessitating the development of new methods to enable the BN modeling of larger multistate flow systems.

Proposed Algorithms for Bayesian Network Modeling of Multistate Systems

Overview

The flowchart of the proposed algorithms is shown in Fig. 2. Each of the modeling and analysis steps is described in detail in the following sections. First, to reduce the size of the CPT, the authors substitute subsystems with components that are either in series or parallel as supercomponents. Next, the authors generate the minimum cut sets (MCSs) of the network to determine the system state for each combination of component states. To facilitate efficiency of the compression algorithm that follows, the authors renumber the supercomponents that now comprise the full system based on two heuristics: whether they may be observed and their appearances in the MCSs. This completes the formulation of the system for the BN model.

Next, as shown in Fig. 2, to construct and perform inference over the model, compression and preprocessing algorithms are proposed. These algorithms are run simultaneously to reduce computational time. To compress the system node CPT, the authors introduce the idea of bundles representing repeated patterns of fixed length in the system state in the CPT. The simultaneous preprocessing of the information in the BN removes the need for storage of intermediate factors of unobserved components during inference, reducing both memory storage requirements and computational time. At the end of the algorithm, the authors obtain a compressed system CPT, $cCPT$, and dictionary of bundles, d, used in the compression. These contain all the information for the BN model of the system compressed in a lossless manner and without making any approximations. The prior probability distribution for the system before any observation is made, λ_1, and subsequent distribution and dictionary, λ_{i+1} and d_{i+1}, respectively, are also obtained for inference on posterior probability distributions given observations on the system. Each of the elements of the proposed method is now described in detail.

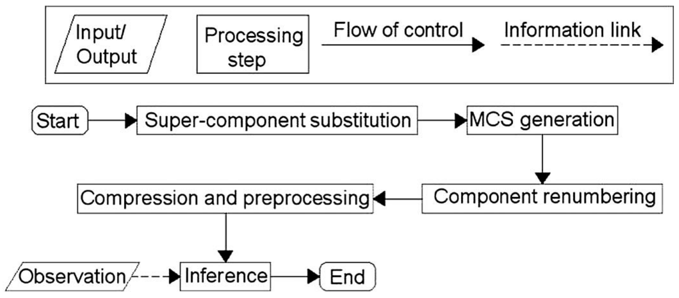

Fig. 2. Flowchart of proposed algorithms for BN modeling of multistate systems

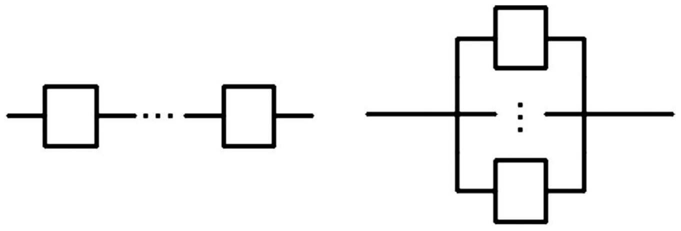

Fig. 3. Supercomponent configurations

Supercomponent (C_c) Substitution

The total number of rows in the CPT for the system node in the BN as shown in Fig. 1, representing, e.g., the performance of an infrastructure system of n components, increases exponentially with n. Decreasing n will significantly reduce the size and computational requirements for the CPT. A common practice in the field of reliability analysis is to combine components in similar configurations into a single component, which the authors call a supercomponent, denoted as C_c. The probability distribution of the ith supercomponent C_{ci} is denoted as $p(C_{ci})$. The two most recognized configurations are shown in Fig. 3 with components in series (left) or parallel (right) subsystems. These simple configurations are widely used in civil infrastructure systems, including redundant components arranged in parallel in mesh-type networks and components in series in linear pipeline systems. Instead of including all individual components in a series or parallel subsystem in the CPT, the authors use one supercomponent with updated probability distributions to represent the subsystem. For consistency in the formulation, a component that does not belong to a series or parallel subsystem is treated as a supercomponent on its own. The probability distribution of a supercomponent can be easily determined by the characteristic of a series or parallel system. In this paper, the authors continually reduce the complexity of the network using supercomponents until no two components in the system remain in series or parallel.

One advantage of using supercomponents is that if additional components are added to the structure of previously defined supercomponents, this will not increase the size of the overall problem because the number of supercomponents is not changed. Only the probability distribution of that supercomponent need be modified. Of course, more complicatedly connected subsystems other than series and parallel configurations can be defined as supercomponents. This may further simplify the structure of the system for specific cases, particularly if there are several repetitions of a similar structure in the network. In this paper, without loss of generality, the authors limit the discussion of supercomponents to series and parallel subsystems.

Minimum Cut Set Generation and Renumbering Components

In the proposed BN formulation, determination of the system state in any row of the CPT is based on the component states and the set of minimum cut sets or minimum link sets (MLSs) of the system. The component states are determined by their row number in the CPT as in Tien and Der Kiureghian (2016). Enumerating all MCSs or MLSs for a system is an nondeterministic polynomial (NP)–time hard problem, though several efficient methods have been developed to do so (Suh and Chang 2000; Li et al. 2007;

Dictionary=

Bundle number	1	2
Bundle	{0,0,0}	{0,1,1}

cCPT=

Bundle number	1	2	1	2	1	2	1	...
# repetitions of bundle	312	2	1	2	1	2	1	...

Fig. 4. Illustration of example three-state system CPT compression

Dictionary=

Bundle number	1	2	3
Bundle	{0,0,0}	{0,1,1}	{0,1,1,0,1,1,0,0,0}

cCPT=

Bundle number	1	3	...
# repetitions of bundle	312	100	...

Fig. 5. Illustration of three-state system CPT compression after combination of phrases

Benaddy and Wakrim 2012). The EG-CUT algorithm proposed by Shin and Koh (1998) generates MCSs for undirected graphs by using a blocking mechanism. Minimum cut sets can be generated at $O(en)$, where e is the number of edges, and n the number of nodes in the graph. Depth-first search-based methods can also be used for MLS generation (Jiang et al., 2016). These have been used to find the MLSs for an infrastructure system of 127 nodes using a personal computer on the order of seconds (Johansen and Tien 2017).

Once the MCSs have been generated, the authors propose two heuristics for numbering the components to further reduce the computational complexity of the problem. The number of a component affects where it appears in the system CPT, i.e., its column in the left-hand side of the CPT as shown in Table 1. The heuristics proposed to renumber the components result in reduced memory storage required for inference and more efficient compression of the initial system CPT. In general, selection of an optimal component numbering in the network is an NP-hard problem (Dechter 1999). In the proposed method, the first priority is to number the components that may be observed to appear as far left in the CPT as possible. When the state of a component is observed, the probability distribution of the system node is updated given that information using Bayes' rule. In most current infrastructure systems, particularly water and gas line infrastructure, not all components have monitoring capabilities. This heuristic takes advantage of this system characteristic to reduce the size of the intermediate probability distributions, denoted λ and called intermediate factors, calculated during the inference process. For the variable elimination inference method used in this paper, when a component is observed, only the part of λ that corresponds with the component being in the observed state need be considered. Numbering the monitored components first to appear on the left side of the CPT reduces the storage required for λ.

To facilitate more efficient compression of the system CPT using the algorithm described in the following section, the second part of the renumbering heuristic is to number components with greater influence in affecting the system state to appear to the left in the CPT (after the monitored components previously described). Appearance in MCSs is used as a proxy for component influence and criticality (Meng 1994). Specifically, after obtaining the MCSs, the authors rank the influence of each component by its number of appearances in a MCS. The most influential component is the one that appears in the most MCSs. By renumbering components based on influence so that more influential components have smaller component numbers results in a CPT where the system state does not change rapidly from row to row in the CPT. This results in a more regular pattern that improves the computational efficiency of compression.

Algorithm for Compressing BN Model

With the BN model created after supercomponent substitution and renumbering, the number of parents of the system node in Fig. 1 is reduced to n_c, where n_c is the number of supercomponents. In constructing the CPT, the authors use the supercomponents instead of original components. Rather than storing all elements in the full CPT, an algorithm to compress the information in the CPT is proposed to reduce the memory storage requirements and make the BN modeling of larger civil infrastructure systems possible. First, the component states in the CPT (left-hand columns in Table 1) need not be stored, as long as they follow a predetermined pattern. Next, compression is accomplished by processing through each row of the full system CPT and storing the values that appear in the system state columns (right-hand columns in Table 1) in a lossless compressed form. To do this, the authors introduce the idea of *bundles*. A bundle is a pattern in the values of the system state of fixed length that is proportional to the number of states. These are more specific than the general *phrases* terminology originally proposed by Ziv and Lempel (1977). However, consistent with previous work, this method stores identified bundles in a *dictionary*. The compressed CPT therefore comprises detected bundles, or patterns of sequences in the system columns of the CPT, which are referenced from the dictionary.

The idea of bundles is important to remove the need to calculate certain *remainder* values when processing through the CPT as in the previously developed algorithms for BN modeling of binary systems. The reader is referred to Tien (2014) for details in the calculation of this remainder. As the number of possible states of the system increases, so does the likelihood of having remainders. Employing bundles removes the possibility of remainders. Let m denote the number of states of the components and system. With fixed-length bundles of lengths that are multiples of m stored in the dictionary rather than allowing phrases of general length, the CPT is thus compressed in groups proportional to m, removing any remainders from the compression process.

The proposed compression algorithm operates as follows. The outputs of the algorithm are the compressed system CPT, $cCPT$, and the accompanying dictionary of bundles, d. Examples of the algorithm are shown in Figs. 4 and 5. The full flowchart of the algorithm is shown in Fig. 6.

For each row $n_r = 1, 2, \ldots, m^{n_c}$ of the system CPT, the states of the supercomponents $S_1, \ldots, S_{n_c}$ represented in that row are computed based on the specific pattern used in defining the CPT. The CPT, as shown in Table 1, is constructed with supercomponents $C_{c1}, \ldots, C_{cn_c}$ organized from left to right. Each row of the CPT is one of the mutually exclusive combinations of component states. The authors determine the state of component $C_{ci}, i = 1, \ldots, n_c$ in row n_r of the CPT according to Eq. (1)

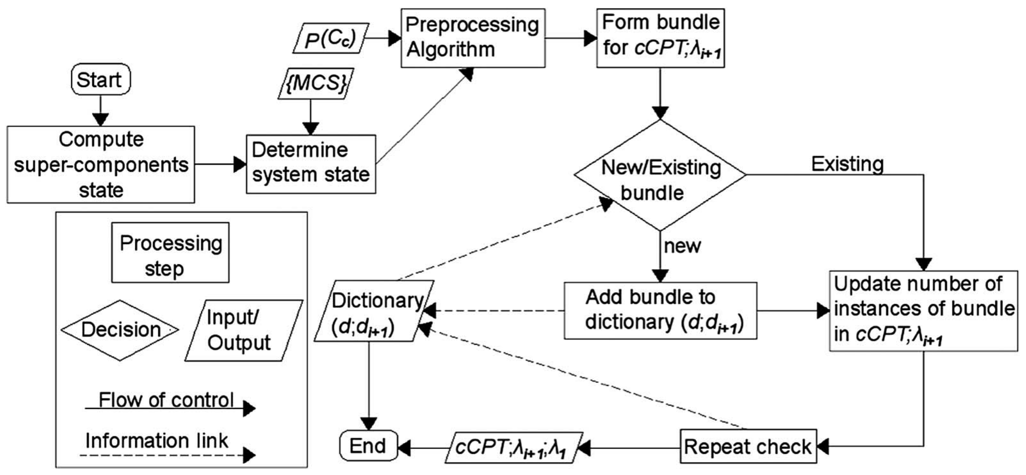
Start
Compute super-components state
Determine system state
$P(C_c)$
{MCS}
Preprocessing Algorithm
Form bundle for cCPT;λ_{i+1}
New/Existing bundle
Existing
new
Add bundle to dictionary (d;d_{i+1})
Update number of instances of bundle in cCPT;λ_{i+1}
Dictionary (d;d_{i+1})
Repeat check
cCPT;λ_{i+1};λ_1
End
Processing step
Decision
Input/ Output
Flow of control
Information link

Fig. 6. Flowchart of compression algorithm

$$S_i = \mathrm{mod}\left[\mathrm{ceil}\left(\frac{n_r}{m^{n_c-i}}\right) - 1, m\right] \qquad (1)$$

where $\mathrm{ceil}(x)$ is the value of x rounded up to the nearest integer; and $\mathrm{mod}(a, m)$ returns the remainder after division of a by m. The possible states of the component $C_{ci} \in \{0, 1, \ldots, m-1\}$.

For each row, the component states are then checked against $\{MCS\}$ (indicating the set of minimum cut sets of the system) by Eq. (2) to determine the state of the system in row n_r, denoted sys_{n_r}

$$sys_{n_r} = \min_{j=1,\ldots,n_{\mathrm{MCS}}} \{\max\{\mathrm{MCS}_{j,n_r}\}\} \qquad (2)$$

where n_{MCS} denotes the number of MCSs of the system; and any one MCS_{j,n_r}, $j = 1, \ldots, n_{\mathrm{MCS}}$ contains the states of the components comprising that MCS in row n_r of the CPT. From the value of sys_{n_r}, the authors obtain the corresponding binary values in the m entries for the system state in row n_r.

In addition to storing phrases as bundles of fixed length proportional to the number of states m, the authors have found that additional computational efficiencies can be achieved by combining multiple bundles. This is particularly effective if there are repeated patterns in the occurrence of bundles. For example, consider a three-state system with dictionary and compressed system node CPT represented by $cCPT$ as shown in Fig. 4. In this case, the pattern in bundles in $cCPT$ is represented by two repetitions of Bundle 2, then 1 repetition of Bundle 1. The memory storage requirements for the compressed CPT can be reduced significantly by defining a new Bundle 3 comprising the bundle $\{0, 1, 1, 0, 1, 1, 0, 0, 0\}$ as shown in Fig. 5. The authors call this new, longer bundle a *clip*. Specifically, the original memory storage requirement is (memory for dictionary) + (memory for $cCPT$) $= (3+3) + (201 \times 2) = 408$ elements, while combining the bundles into a clip results in a memory storage requirement of $(3+3+9) + (2 \times 2) = 19$ elements. Though the size of the dictionary slightly increases with the longer-length clip, the size of the compressed CPT is significantly reduced. Note that combining bundles is not beneficial in all cases. For example, if there are 50 repetitions of Bundle 2 followed by 50 repetitions of Bundle 1, the length of the clip itself would be $(50 + 50) \times 3 = 300$ elements.

To identify repeated patterns among bundles to combine them into clips, the authors process through $cCPT$ from left to right through the columns of $cCPT$. For improved efficiency of repeated pattern finding, the length of the clip was limited to the number of states, i.e., for a three-state system, the authors do not look for a repeated pattern over the length of three. Performing this repeat check multiple times, i.e., combining repeated clips into new, longer clips, was also considered. However, this required additional computational time and did not result in savings in memory storage. Therefore, the repeat check algorithm is run through the data once for full compression.

Algorithm for Preprocessing BN Model

Once the BN has been constructed, inference is required to draw conclusions about the system. In the proposed method, a preprocessing algorithm is used as a prerequisite for inference. This is done simultaneously with the compression algorithm to eliminate the need to run through the data twice and to further reduce memory storage requirements. The preprocessing algorithm is based on the classical variable elimination algorithm (Dechter 1999). In this algorithm, inference is achieved by eliminating all other nodes in the network until arriving at the node of interest. Elimination of each node corresponds to summing of the joint distribution over all states of the node, resulting in an intermediate factor λ that is used during the next step of elimination.

Consider n_c supercomponents of a system $C_{c1}, \ldots, C_{cn_c}$ with probability distributions $p(C_{c1}), \ldots, p(C_{cn_c})$. If the maximum supercomponent number of the monitored components is i, then the intermediate factor of interest is λ_{i+1}. In a backward elimination order of $k = \{n_c, n_{c-1}, \ldots, i+1\}$, the elimination of a component k results in an intermediate factor λ_k. At each elimination step, the authors multiply the values of the elements in λ_k with the probability distribution of the eliminating component to obtain λ_{k-1} in the next step. This algorithm results in exact inference over the network.

The key to computationally tractable exact inference in BNs with many parent nodes using the proposed compression approach is that the intermediate factor λ_{i+1} is also stored in compressed form following the same methodology used to create $cCPT$. This is done using the proposed preprocessing algorithm that is run simultaneously with the compression algorithm. Preprocessing is possible because the full intermediate factor does not have to be constructed for the variable elimination method to proceed. For example, as the authors process through the CPT from the first row to the mth row, the first entry of λ_{n_c} is already determined. The following rows will not affect the value of that first entry and after calculating the value, the first mth entries can be cleared from storage. The authors need not present at any time the full λ_{n_c}. Similarly, once the mth entry is stored in λ_k, the next entry for λ_{k-1} can be determined. Because the intermediate factor of interest is λ_{i+1}, for all other intermediate factors, storage of only m entries for each λ is needed at a time.

Let e_k denote the values currently stored in λ_k. When the n_rth row is processed, the updating rule for intermediate factors λ_k, $k > i+1$ is to use the m entries in e_k and $p(C_{ck-1})$ to create the next entry for e_{k-1}. This is done with the calculation shown in Eq. (3)

$$e_k' = e_k[P(C_{ck}=0) \quad \cdots \quad P(C_{ck}=m-1)]^T \tag{3}$$

where e_k' indicates the next entry for λ_{k-1}. The contents in e_k can now be cleared. Once the authors arrive at λ_{i+1}, all entries are stored for future use in inference using the compression methodology. Thus, CPT and λ_{i+1} are compressed simultaneously. The algorithms operate as follows.

Preprocessing Algorithm

```
Input: $n_c$, $m$, $p(C_c)$, $i$
Output: $\lambda_{i+1}$, $d_{i+1}$, $\lambda_1$
For $n_r$ from 1 to $m^{n_c}$
Determine system state for row number $n_r$ by Eqs. (1) and (2)
Set $k = n_c$, $N = \text{floor}(\frac{n_r}{m})$, $R = \text{mod}(n_r, m)$
While $R = 0$
Update $e_k$ to $e_k'$ based on $p(C_{ck})$ from $p(C_c)$ by Eq. (3).
Update contents in $e_k$ and $e_{k-1}$.
Update $k = k-1$, $N = \text{floor}(\frac{N}{m})$, $R = \text{mod}(N, m)$
End while
Compress entries in $\lambda_{i+1}$ and companion dictionary $d_{i+1}$ using the compression algorithm
End for
                $p(sys) = \lambda_1$
```

Compression Algorithm

```
Input: $n_c$, $m$, $\{MCS\}$, $p(C_c)$, $i$
Output: $cCPT$, $d$, $\lambda_{i+1}$, $d_{i+1}$, $\lambda_1$
For $n_r \leftarrow 1$ to $m^{n_c}$, do as shown in Fig. 6
```

Test Application

The proposed algorithms are now applied to a test application to illustrate their use. The example system is adopted from Bensi et al. (2013) with added complexity to demonstrate the methodology. The system has previously been used as an example of an infrastructure distribution network, providing a resource (e.g., water or gas) from a source to a sink as in Tien and Der Kiureghian (2015). In this paper, the authors begin with this network as shown in Fig. 7. However, because of the combination of series and parallel configurations of components C_1–C_8, it can be reduced into a single supercomponent. Therefore, the authors call this system supercomponent C_{ci} and increase the complexity of the system to form the network shown in Fig. 8. This network is formed as a combination of subsystems. Representing the network in Fig. 8 using supercomponents results in the system shown in Fig. 9, with each supercomponent $C_{ci}, i = 1, \ldots, 7$ as shown in Fig. 7. Thus, the full network of 56 components is represented with 7 supercomponents. Compared to the system in Fig. 7 that can be reduced to one supercomponent, the network shown in Fig. 9 is irreducible using the described supercomponent methodology. Therefore, this network is chosen to test performance of the algorithms.

Suppose all components are independent and can be in one of five possible states modeling the level of flow compared to maximum flow capacity. Let state $S_i = \{0, 1, 2, 3, 4\}$ denote flow = 0, 25, 50, 75, and 100% of maximum capacity. Prior probability distributions for each component are listed in Table 2, where S_i denotes the state of component C_i. The resulting prior distribution for each supercomponent C_{ci} is listed in Table 3. Suppose components

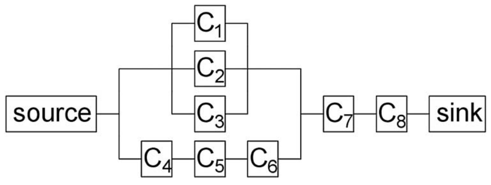

Fig. 7. Supercomponent C_{ci} configuration

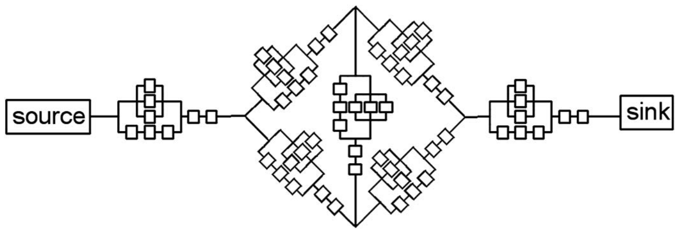

Fig. 8. Example system

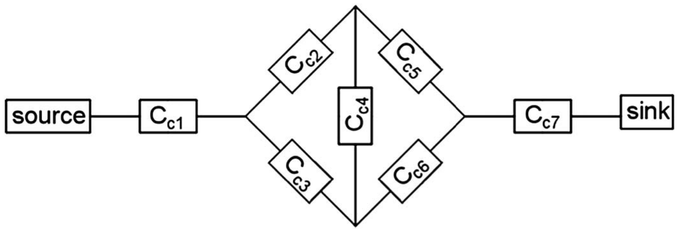

Fig. 9. Example system with supercomponent representation

Table 2. Prior Probability Distributions for Components Constituting Supercomponent C_{ci}

S_i	$p(C_1)$	$p(C_2)$	$p(C_3)$	$p(C_4)$	$p(C_5)$	$p(C_6)$	$p(C_7)$	$p(C_8)$
0	0.18	0.16	0.14	0.12	0.10	0.08	0.06	0.04
1	0.19	0.18	0.17	0.16	0.15	0.14	0.13	0.12
2	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
3	0.21	0.22	0.23	0.24	0.25	0.26	0.27	0.28
4	0.22	0.24	0.26	0.28	0.30	0.32	0.34	0.36

Table 3. Prior Probability Distribution for Supercomponent C_{ci}

S_i	$p(C_{ci})$
0	0.0986
1	0.2364
2	0.3257
3	0.2692
4	0.0701

Table 4. Dictionary for *cCPT*: *d*

Bundle number	Bundle
1	$(0,0,0,0,0)$
2	$(1,1,1,1,1)$
3	$(1,0,0,0,0)$
4	$(1{,}0,0{,}0,0{,}1,0{,}0,\ \ldots,1{,}1,1{,}1,1)$
5	$(1{,}0,0{,}0,0{,}1,0{,}0,\ \ldots,1{,}1,1{,}1,1)$

Table 5. Bundle Number and Repetitions for *cCPT*

Bundle number	Number of repetitions of bundle
2	3,251
5	4
4	5
5	4
...	...
2	1
5	23
3	24

Table 6. Dictionary for $c\lambda_3$: d_3

Bundle number	Bundle
1	$(1{,}1,1{,}1,1)$
2	$(0.2025,0.1081,0.1081,0.1081,0.1081)$

C_1 of C_{c1} and C_{c2} are instrumented and can be monitored for updating of the system state. For the reduced system shown in Fig. 9, the set of MCSs is $\{MCS\} = \{(C_{c1}), (C_{c7}), (C_{c2}, C_{c3}), (C_{c5}, C_{c6}), (C_{c2}, C_{c4}, C_{c6}), (C_{c3}, C_{c4}, C_{c5})\}$.

Results of Implementing Compression and Preprocessing Algorithms

To implement the compression and preprocessing algorithms, for each row in the system CPT, the states of the components are first determined by Eq. (1). The state of the system is then found using MCSs as shown in Eq. (2). The resulting dictionary, d, and compressed system CPT, *cCPT*, obtained after implementing the compression algorithm are shown in Tables 4 and 5, respectively. After compression, there are a total of five bundles identified and *cCPT* contains 120 columns. Note that Bundle 4 is a clip combining four repetitions of Bundle 3 and one repetition of Bundle 2. Bundle 5 combines 24 repetitions of Bundle 3 and one repetition of Bundle 2. Compared to 78,125 rows of the full CPT, the compressed form comprising 2025 total elements achieves orders of magnitude savings in memory storage for the CPT. This compression is done in a lossless manner and without making any approximations.

While the compression algorithm is implemented for the system CPT, $P(sys = 0) = 0.2045$ is calculated simultaneously using the preprocessing algorithm. The probability distribution over the system state will be updated if the monitored components are identified to be in a specific state. In this case, if components C_1 of C_{c1} and C_{c2} are monitored, then the intermediate factor of interest is λ_3. This is created simultaneously using the preprocessing algorithm as the authors compress the original CPT. The d_3 and the compressed λ_3 denoted $c\lambda_3$ are as listed in Tables 6 and 7.

Table 7. Bundle Number and Repetitions for $c\boldsymbol{\lambda}_3$

Bundle number	Number of repetitions of bundle
1	1
2	4

Inference

Once d_3 and $c\lambda_3$ are obtained, it is possible to conduct exact inference over the network. Suppose that C_1 in both C_{c1} and C_{c2} are in State 0. The authors then update the prior distribution of C_1 as $P(C_1 = 0) = 1$. For all other states $m = 1, \ldots, 4$, $P(C_1 = m) = 0$. As a result, the probability distribution for supercomponents C_{c1} and C_{c2} are updated as: $P(C_c = 0) = 0.1031$, $P(C_c = 1) = 0.2580$, $P(C_c = 2) = 0.3382$, $P(C_c = 3) = 0.2453$, and $P(C_c = 4) = 0.0554$. Then the new C_c distributions are used with λ_3 to obtain the updated system state distribution $P(sys = 0|C_1 \,\text{in}\, C_{C1}$ and $C_{C2} = 0) = 0.2088$. This represents a slight increase in the probability of failure from 0.2045. This is attributable to C_1 being one component of a parallel subsystem within only two supercomponents. In general, once the compressed $c\lambda_{i+1}$ is obtained from the preprocessing algorithm, the inference is computed using simple calculations based on $c\lambda_{i+1}$.

In many system reliability problems, an observation at the system level is made, and the objective is to identify the components most likely to have led to that system behavior. This is called backward inference. Part of the power of BNs is in its ability to perform backward inference by Bayes' rule: $p(C|C_{ci}) = p(C_{ci}|C)p(C)/p(C_{ci})$ and $p(C_{ci}|sys) = p(sys|C_{ci})p(C_{ci})/p(sys)$. An example of the results of this inference is given in Fig. 10, which shows the updated probability distributions of each component being in different states given a supercomponent C_{ci} being in State 2. Fig. 11 shows the updated probability distributions of each supercomponent being in different states given the system being in State 2. When researchers have evidence of underperformance at the system level, the results in Fig. 10 provide information on the importance of individual supercomponents comprising the system. Using the chain rule and total probability, the influence of specific components constituting the supercomponents is determined by $p(C|sys) = \sum_{C_{ci}} p(C|C_{ci}, sys)p(C_{ci}|sys) = \sum_{C_{ci}} p(C|C_{ci})p(C_{ci}|sys)$. These inference results enable identification of critical components in the system to inform decision making in the maintenance and reinforcement of the critical components to minimize risk of system underperformance.

Algorithm Performance: Memory Storage

In the previous section, the authors applied the proposed algorithms to an example system to demonstrate their use. As the objective of the algorithms is to reduce the memory storage requirements and increase the computational efficiency of BN modeling of multistate flow networks as the number of components in the system increases, the performance of the proposed algorithms is now compared with existing methods for modeling systems of increasing size.

First, the performance is assessed in terms of memory storage. The existing method for comparison is the junction tree (JT) algorithm as implemented in the Bayes Net Toolbox in *MATLAB*. The JT algorithm is generally known for its efficiency in performing exact inference for graphical networks (Spiegelhalter et al 1993). The authors increase the size of the system by adding to the last supercomponent components in parallel up to a total number of supercomponents n as shown in Fig. 12. As the purpose is to

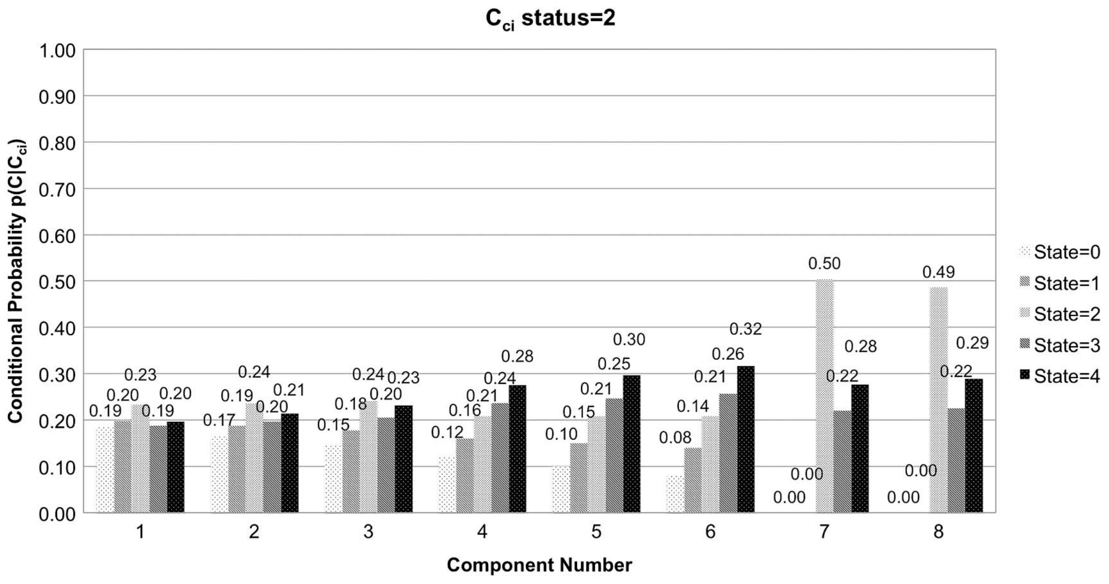

Fig. 10. Updated component probability distributions given a supercomponent C_c in State 2

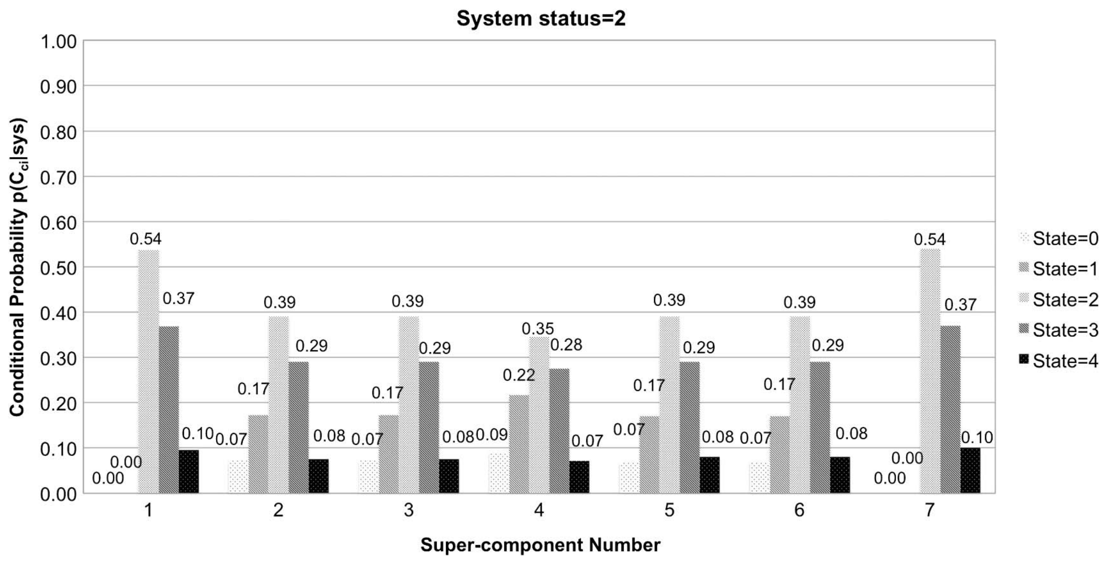

Fig. 11. Updated supercomponent probability distributions given system in State 2

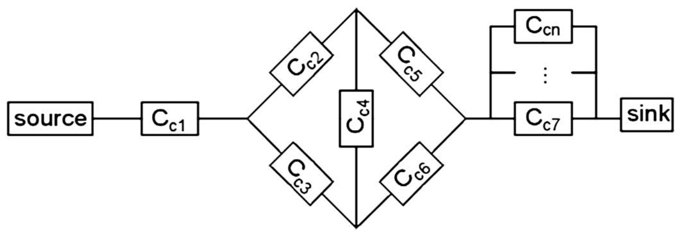

Fig. 12. Expanded example system

analyze the effect on compression of increasing the number of components, supercomponents C_{c7} to C_{cn} are not combined into a single supercomponent. Similar analyses can be conducted for increasing the number of components elsewhere in the system.

The maximum number of elements required to be stored to create the BN model and for inference is used as a proxy for memory storage demand. Results are based on the performance of the algorithms run in *MATLAB* on a 16 GB RAM computer. In both methods, supercomponents are employed for a fair comparison. For the system in Fig. 12, when n reaches 12, the storage demand for the JT algorithm exceeds memory. Results for the JT compared to the proposed algorithms as the number of supercomponents in the system increases is shown in Fig. 13. Table 8 lists these values and computes the data compression ratio of the proposed algorithms, i.e., the ratio of the number of elements required for the JT compared to proposed algorithms.

In Fig. 13 and Table 8, the values being recorded are the maximum number of elements stored during construction of and inference over the BN for systems of increasing size. For the proposed algorithms, the value includes both the elements in compressed intermediate factors $c\lambda_k$ and the bundles in the dictionaries d_k used in

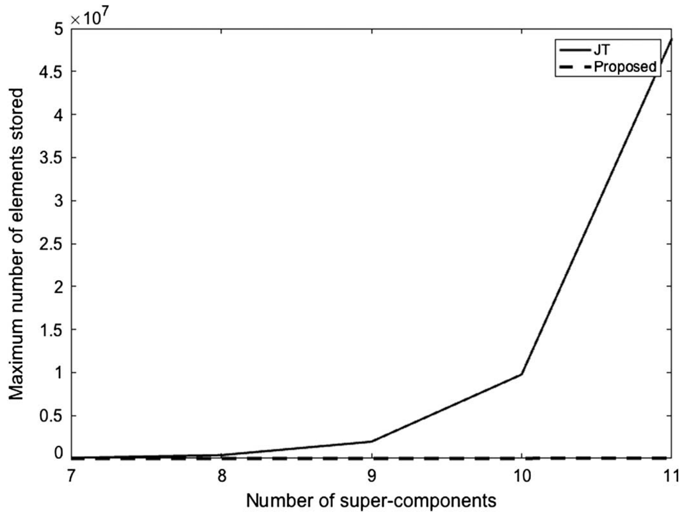

Fig. 13. Memory storage requirements of JT compared to proposed algorithms

defining $c\lambda_k$. It is shown in Fig. 13 that the proposed algorithms achieve significant savings in the memory storage requirement for the BN model. The maximum number of elements stored is orders of magnitude smaller than required for the JT algorithm. For this example, the memory storage in both cases increases exponentially with system size. However, the base of the increase for the proposed algorithms is approximately 2 compared to 5 for JT. For the 11 supercomponents case, which represents a system of 88 components, the proposed algorithms require 26,580 elements to be stored. This is compared to $5^{88} = 3.23 \times 10^{61}$ elements required for the original formulation.

For the JT algorithm, when the number of supercomponents in the system increases to 12, the size of the CPT exceeds the available memory storage capacity and the BN model cannot be constructed. Further, the last row in Table 8 shows the number of times by which the data has been compressed, i.e., the data compression ratio, using the proposed algorithms. Note again that the compression is lossless, so the authors are not losing any information nor are they making any approximations during the compression and inference processes. From the results, it is shown that as the size of the system increases, so does the compression efficiency of the proposed algorithms, with an exponentially increasing data compression ratio.

Algorithm Performance: Computational Efficiency

Now the performance of the proposed algorithms is examined in terms of computational efficiency. Table 9 lists the computation time needed for calculation over the network for JT compared to proposed algorithms. The JT algorithm time includes constructing the full CPT and calculating the prior probability distribution $p(sys)$. The proposed algorithms time includes computation required for both compression and preprocessing information in the BN.

Table 9. Comparison of Computation Time for Calculation for JT versus Proposed Algorithms (s)

	Number of supercomponents				
Methods	7	8	9	10	11
JT	24.01	131.29	710.01	3,564.23	17,563.11
Proposed	35.00	188.68	1,077.10	6,273.59	26,519.32
Proposed/JT	1.46	1.44	1.52	1.76	1.51

From Table 9, it is shown that the computation times increase as the size of the system increases, a well-known example of the problem of dimensionality. The time required for the proposed algorithm is approximately 1.5 times that of the JT algorithm. However, with the large savings in memory storage, the authors consider the performance metrics for the proposed algorithms an acceptable tradeoff. For example, for a system of 11 supercomponents, the data is compressed by 1,837 times while the computation time increases by 1.51 times.

Memory compared to computation time requirements are also fundamentally different. Memory storage is a hard constraint. If the maximum required memory exceeds storage capacity of a program or machine, no further analysis can be performed. While it is true that memory can be distributed, e.g., in cloud storage, a hard limit still exists. In contrast, computation time is more flexible. Indeed, methods such as parallel computing exist to address computation time. Further, for the proposed algorithms, the compression efficiency grows significantly as the number of components increases, while the computation time ratio remains stable. Thus, the algorithms enable larger multistate flow systems to be modeled using BNs than previously possible for probabilistic inference and reliability assessment.

Conclusion

In this paper, the authors propose new algorithms for constructing and performing inference in BN models for reliability assessment of multistate infrastructure flow networks. The new algorithms address the major system size limitation in the use of BNs for modeling large systems and the increased complexity of modeling multistate flow compared to binary connectivity networks. They include a supercomponent substitution method, which reduces the total number of nodes in the BN, and component renumbering heuristics to increase computational efficiency. Algorithms to losslessly compress the system CPT while simultaneously calculating and compressing intermediate factors for exact inference are then proposed. The performance of the proposed algorithms is tested using an example system. Compared with existing methods, the new algorithms achieve orders of magnitude savings in memory storage. This is accompanied by a slight decrease in computational efficiency. However, the compression efficiency improves with an exponentially increasing data compression ratio, while the computation time ratio remains stable as the size of the system increases. Together these algorithms enable multistate flow networks to be

Table 8. Comparison of Memory Storage Required for JT versus Proposed Algorithms

	Number of supercomponents				
Methods	7	8	9	10	11
JT	78,125	390,625	1,953,125	9,765,625	48,828,125
Proposed	2,025	11,080	11,580	14,080	26,580
Data compression ratio	38.58	35.26	168.66	693.58	1,837.03

modeled as BNs. As infrastructures age and are subjected to increasing hazards, the use of BNs for assessment over a variety of scenarios, including updating with new information, enables prioritization of components and decision making across the network to increase the reliability of these critical systems.

Acknowledgments

Support for this work by the National Science Foundation through Grant No. CNS-1541074 is acknowledged.

References

Benaddy, M., and Wakrim, M. (2012). "Cutset enumerating and network reliability computing by a new recursive algorithm and inclusion exclusion principle." *Int. J. Computer Appl.*, 45(16), 22–25.

Bensi, M., Der Kiureghian, A., and Straub, D. (2013). "Efficient Bayesian network modeling of systems." *Reliab. Eng. Syst. Saf.*, 112, 200–213.

Bobbio, A., Portinale, L, Minichino, M., and Ciancamerla, E. (2001). "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks." *Reliab. Eng. Syst. Saf.*, 71(3), 249–260.

Boudali, H., and Dugan, J. B. (2005). "A discrete-time Bayesian network reliability modeling and analysis framework." *Reliab. Eng. Syst. Saf.*, 87(3), 337–349.

Bouissou, M., and Pourret, O. (2003). "A Bayesian belief network based method for performance evaluation and troubleshooting of multistate systems." *Int. J. Reliab. Qual. Saf. Eng.*, 10(4), 407–416.

Dechter, R. (1999). "Bucket elimination: A unifying framework for reasoning." *Artif. Intell.*, 113(1–2), 41–85.

Gu, Y. K., and Yang, Z. X. (2013). "Reliability analysis of multi-state systems based on Bayesian network." *2013 Int. Conf. on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, IEEE, Piscataway, NJ, 332–336.

Jensen, F. V., and Nielsen, T. D. (2007). *Bayesian networks and decision graphs*, 2nd Ed., Springer, New York.

Jiang, X., Bai, R., Atkin, J., and Kendall, G. (2016). "A scheme for determining vehicle routes based on arc-based service network design." *Inf. Syst. Oper. Res.*, 55(1), 16–37.

Johansen, C., Horney, J., and Tien, I. (2016). "Metrics for evaluating and improving community resilience." *J. Infrastruct. Syst*, 10.1061/(ASCE)IS.1943-555X.0000329, 04016032.

Johansen, C., and Tien, I. (2017). "Probabilistic multi-scale modeling of interdependencies between critical infrastructure systems for resilience." *Sustainable Resilient Infrastruct.*, in press.

Khakzad, N., Khan, F., and Amyotte, P. (2011). "Safety analysis in process facilities: Comparison of fault tree and Bayesian network approaches." *Reliab. Eng. Syst. Saf.*, 96(8), 925–932.

Kim, M. C. (2011). "Reliability block diagram with general gates and its application to system reliability analysis." *Ann. Nucl. Energy*, 38(11), 2456–2461.

Li, J., Qian, Y., and Liu, W. (2007). "Minimal cut-based recursive decomposition algorithm for seismic reliability evaluation of lifeline networks." *Earthquake Eng. Eng. Vib.*, 6(1), 21–28.

Mahadevan, S., Zhang, R., and Smith, N. (2001). "Bayesian networks for system reliability reassessment." *Struct. Saf.*, 23(3), 231–251.

MATLAB [Computer software]. MathWorks, Natick, MA.

Meng, F. C. (1994). "Comparing criticality of nodes via minimal cut (path) sets for coherent systems." *Probab. Eng. Inf. Sci.*, 8(1), 79–87.

Shin, Y. Y., and Koh, J. S. (1998). "An algorithm for generating minimal cutsets of undirected graphs." *Korean J. Comput. Appl. Math.*, 5(3), 681–693.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L., and Cowell, R. G. (1993). "Bayesian analysis in expert systems." *Stat. Sci.*, 8(3), 219–247.

Suh, H., and Chang, C. K. (2000). "Algorithms for the minimal cutsets enumeration of networks by graph search and branch addition." *Proc., 25th Annual IEEE Conf. on Local Computer Networks*, IEEE, Piscataway, NJ, 100–110.

Tien, I. (2014). "Bayesian network methods for modeling and reliability assessment of infrastructure systems." Ph.D. thesis, Univ. of California, Berkeley, CA.

Tien, I., and Der Kiureghian, A. (2013). "Compression algorithm for Bayesian network modeling of binary systems." *Safety, reliability, risk and life-cycle performance of structures and infrastructures*, G. Deodatis, B. Ellingwood, and D. Frangopol, eds., CRC Press, New York, 3075–3081.

Tien, I., and Der Kiureghian, A. (2015). "Compression and inference algorithms for Bayesian network modeling of infrastructure systems." *Proc., 12th Int. Conf. on Applications of Statistics and Probability in Civil Engineering*, T. Haukaas, ed., Vancouver, Canada.

Tien, I., and Der Kiureghian, A. (2016). "Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems." *Reliab. Eng. Syst. Saf.*, 156, 134–147.

Tien, I., and Der Kiureghian, A. (2017). "Reliability assessment of critical infrastructure using Bayesian network." *J. Infrastruct. Syst.*, in press.

Tong, Y., and Tien, I. (2016). "Algorithms for Bayesian network modeling of multi-state infrastructure flow systems." *Engineering Mechanics Institute and Probabilistic Mechanics and Reliability Conf.*, Nashville, TN.

Torres-Toledano, J. G., and Succar, L. E. (1998). *Bayesian networks for reliability analysis of complex systems*, Springer, Berlin, 195–206.

Yeh, W. C. (2013). "Evaluating the reliability of a novel deterioration-effect multi-state flow network." *Inf. Sci.*, 243, 75–85.

Ziv, J., and Lempel, A. (1977). "A universal algorithm for sequential data compression." *IEEE Trans Inf. Theory*, 23(3), 337–343.