Reliability Assessment of Critical Infrastructure Using Bayesian Networks

Iris Tien, Ph.D., A.M.ASCE[1]; and Armen Der Kiureghian, Ph.D., M.ASCE[2]

Abstract: The authors present a Bayesian network (BN)-based approach for modeling and reliability assessment of infrastructure systems. The BN is a powerful framework that is able to account for uncertainties in component and system parameters, and perform updating of system assessments with new information. The exponential increase in memory storage required for the BN model as the size of the system increases has limited the applicability of BNs for reliability assessment of large infrastructure systems. Recently, a data-compression method was proposed to address this limitation. While significantly reducing the memory storage, computational time for constructing the BN and performing inference increased. In this paper, new methodologies are developed to increase the computational efficiency of a compression-based approach for BN modeling and reliability assessment of infrastructure systems. These include algorithms to improve the computational efficiency of the initial compression for constructing the BN, subsequent inference over the network, and overall system formulation. The algorithms are applied to a test example system to examine their performance for systems of increasing size, as well as to a 59-component power distribution network to demonstrate application to real systems. Performance of the proposed methodologies is compared to that of an existing, widely used BN algorithm. With the heuristics employed, the new algorithms are shown to achieve significant gains in both memory storage and computation time, enabling the modeling of large infrastructure systems as BNs for system reliability analysis. **DOI:** 10.1061/(ASCE)IS.1943-555X.0000384. *© 2017 American Society of Civil Engineers.*

Introduction

Infrastructure systems are complex, comprised of many interconnected, interacting components. Ensuring the continued functioning of these systems in an uncertain and dynamically changing environment is critical to the health, security, and growth of our communities (Johansen et al. 2017). In this paper, the authors present a Bayesian network–based approach for modeling and reliability assessment of infrastructure systems. Bayesian networks (BNs) are particularly well suited for reliability assessment of critical infrastructures because of their ability to account for uncertainties in component and system performance, as well as for their ease in updating system state assessments in light of new information. In addition, their transparent graphical nature facilitates use by nonexperts in systems modeling and probabilistic analysis.

Previous studies of BN modeling for infrastructure systems, however, have been limited by the size and complexity of the system that can be tractably modeled as a BN. Specifically, exponentially increasing memory demand as the size of the system increases quickly renders the model intractable. To address this limitation, the authors previously developed new methodologies to enable the BN modeling of larger infrastructure systems than is possible with existing methods (Tien and Der Kiureghian 2016). Compression techniques were utilized to store the required probabilistic data and new inference algorithms were developed that directly utilize the compressed data without decompressing or recompressing them, thus significantly reducing the requirements for memory storage. However, these techniques, while drastically reducing the memory demand, increase the required computation time. In this paper, the authors enhance the proposed BN methodology for modeling and reliability assessment of infrastructure systems by developing several heuristics and corresponding algorithms that significantly improve the computational efficiency of the method. The gained computational advantage is demonstrated through a controlled example system with variable size. The methodology is further illustrated by its application to a 59-component power distribution network.

The rest of the paper is organized as follows: The authors first provide a brief background on BNs and their use for reliability assessment of critical infrastructure systems. Next, the authors describe their compression-based approach for BN modeling of larger systems. The authors describe three heuristics and corresponding algorithms—algorithms for compression, inference, and supercomponents—that are constructed to significantly improve the computational efficiency of the proposed method to enable application to reliability assessment of infrastructure systems. The authors demonstrate the computational gain achieved by these new algorithms by examining a test example with varying size. Finally, the authors analyze a 59-component power distribution network to demonstrate application of the new methodology to real systems.

Bayesian Networks and Their Use in Modeling Infrastructure Systems

A Bayesian network is a probabilistic graph comprised of nodes and links. Each node represents a random variable and each link describes the probabilistic dependency among the variables it connects. Fig. 1 shows a BN model of an infrastructure system, where nodes $C_1, \ldots, C_n$ represent the states of components and the node *sys* represents the state of the system. The state of the system depends on the states of its constituent components. Therefore, there

[1]Assistant Professor, School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0355 (corresponding author). E-mail: itien@ce.gatech.edu

[2]Taisei Professor of Civil Engineering Emeritus, Dept. of Civil and Environmental Engineering, Univ. of California, Berkeley, CA 94720-1710; President, American Univ. of Armenia, Yerevan 0019, Armenia.

Note. This manuscript was submitted on August 9, 2016; approved on March 27, 2017; published online on July 8, 2017. Discussion period open until December 8, 2017; separate discussions must be submitted for individual papers. This paper is part of the ***Journal of Infrastructure Systems***, © ASCE, ISSN 1076-0342.

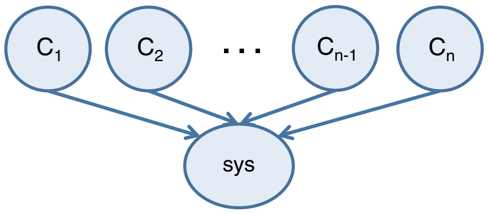

Fig. 1. BN model of an infrastructure system comprised of n components

exists a link in the BN representing the dependency between every component and the system node as shown in Fig. 1.

The BN formulation enables infrastructure system assessments accounting for the probabilistic relationships between component and system states. Extension to the modeling of multiple interdependent infrastructure systems is also possible (Johansen and Tien 2017). In addition, when new information about the system is available, e.g., that a particular component is in the failed state, information entered into any node propagates throughout the network to update reliability assessments at all nodes. This feature of BNs is particularly useful for probabilistic assessment of an infrastructure system in an environment of evolving information, such as that in the aftermath of a natural hazard as observations about the states of system components are gradually made.

The BN as shown in Fig. 1, however, requires an exponentially increasing amount of memory storage to construct as the number of components in the system increases. This is attributable to the conditional probability table (CPT) that must be associated with the system node in the BN. The CPT for a given node defines the probability distribution of the variable represented by that node given each of the mutually exclusive combinations of the states of the parent nodes on which it depends. Consider a binary system with n components, where the components and system can be in one of two states: the functioning state indicated by the numeral 1, and the failure state indicated by the numeral 0. The CPT for the system node is then as shown in Table 1, where each row in the CPT gives the state of the system in the last column given the states of the components in the preceding columns. The CPT consists only of 0's and 1's because it is assumed that, given the states of the components, the system state is known with certainty.

In general, it is only necessary to store the last column of the CPT since the order of component states can be preselected (Tien and Der Kiureghian 2015). For the binary system, the rightmost column in the CPT consists of 2^n elements. Thus, the size of the CPT of the system node increases exponentially as the number of components in the system increases. As an example, modeling an infrastructure system comprised of 100 components requires storage of a vector of length 1.27×10^{30} to build the BN. This clearly poses a significant computational challenge, particularly for assessment of infrastructure networks, which are typically large and comprised of many components.

Table 1. Example Conditional Probability Table for System Nodes

C_1	⋯	C_{n-1}	C_n	sys
0	⋯	0	0	0
0	⋯	0	1	0
0	⋯	1	0	1
0	⋯	1	1	0
⋮	⋮	⋮	⋮	⋮
1	⋯	0	0	0
1	⋯	0	1	1
1	⋯	1	0	1
1	⋯	1	1	1

The approach described above can be extended to systems with multistate components, as long as the system state is described in a binary sense. For example, consider a flow system, where each component has multiple states of flow capacity and the system node describes the flow capacity at one or more critical nodes. If the system is defined as having failed if the flow capacity at any of the critical nodes is below a safe threshold, then the system state is either functioning (numeral 1) or failed (numeral 0). Hence, the system column of the CPT again consists of 0's and 1's (Tong and Tien 2017), and the methods described in this paper can be used. In spite of this generality, in the remainder of this paper the authors focus only on systems with binary components. Furthermore, the scope of this paper is limited to systems with statistically independent component states. For the case of dependent components, parent variable updating and initial elimination of parent nodes must be performed, as in Tien and Der Kiureghian (2016). The reader is referred to that paper for the details.

BNs for Reliability Assessment of Critical Infrastructure Systems

Despite the utility of BNs for reliability assessment and updating, previous studies using BNs for infrastructure systems are limited. The infrastructure systems analyzed are often simplified to accommodate the computational limitations of the BN framework. For example, the power systems studied by Di Giorgio and Liberati (2011, 2012) were modeled as cascaded systems, where the nodes in each layer of the system depend only on the states of at most two parent nodes. In reality, many infrastructure systems do not operate as cascades; instead, failures can happen simultaneously across the network. Jha (2009) used BNs to study critical transportation infrastructure. However, the BN only included nodes for general risk variables, rather than component nodes to investigate the effect of individual component performance on system outcomes or to assess reliability across an infrastructure network.

BNs have been used for the modeling and reliability assessment of general systems. These studies, however, are limited in the sizes of the systems modeled. For example, Torres-Toledano and Succar (1998), Nielsen et al. (2000), Mahadevan et al. (2001), Bobbio et al. (2001), and Boudali and Dugan (2005) analyzed systems of 5, 7, 8, 10, and 16 components, respectively. BN models of systems of this size are not sufficient for meaningful studies of infrastructure systems. Even for these relatively small systems, simplifying assumptions have been employed, including a single-fault failure assumption by Nielsen et al. (2000), and a *branch and bound* approach by Mahadevan et al. (2001) to discard events of relatively low probability to reduce computational demand.

Bensi et al. (2013) proposed a method to address the system size limitation in BNs by using a topology optimization approach. This creates more efficient chainlike BNs rather than models with a converging structure as shown in Fig. 1. However, the proposed optimization program must consider all permutations of component indices in the network and, therefore, may itself become intractably large for large systems. Finally, Di Giorgio and Liberati (2012) used BNs to analyze an electrical control system. However, extensive simulation and sampling was used for inference in the study because, as the authors state, "Computational resources were not sufficient to perform an exact belief update."

Sampling-based methods run many simulations over the network to converge to a solution, while other approximate approaches iteratively or recursively calculate narrowing bounds for system reliability. In both cases, computational requirements may exceed what is feasible for an infrastructure reliability assessment, particularly if real-time updating of system assessments in light of new information is desired. In contrast, the authors focus on exact inference methods in this study. The algorithms and heuristics presented in the following sections do not make any approximations and provide exact inference calculations across the network. The authors' method enables direct computation of measures of reliability, e.g., exact values of probabilities of failure of systems or components, including conditional probabilities of failure based on current component states.

In recent studies (Tien 2014; Tien and Der Kiureghian 2016), the authors proposed a compression algorithm to store the system CPT in a compact form to reduce the memory demand. Additional algorithms were developed to perform inference in the BN without decompressing or recompressing the CPT and other intermediate data tables. As mentioned in the Introduction, this approach drastically reduces the memory requirements. However, this improvement is achieved at the expense of additional computation time. In this paper, the authors present three heuristics and corresponding algorithms that significantly reduce the computation time of the compression-based approach without increasing the memory demand. These enable the use of the proposed algorithms for BN modeling and reliability assessment of infrastructure systems.

Proposed Methodologies for BN Modeling of Infrastructure Systems

Algorithm for Computationally Efficient Compression

The concept of compression is to compress the system column in the CPT, which is exponentially increasing in length, in a lossless manner such that memory storage is reduced without loss of information or making any approximations. As shown in Table 1, the values in the system column in the CPT are either 0 or 1. The original compression algorithm, which integrates the classical compression techniques of run-length encoding [E. L. Hauck, "Data compression using run length encoding and statistical encoding," U.S. Patent No. 4,626,829A (1986)] and the Lempel-Ziv algorithm (Ziv and Lempel 1977), takes advantage of this property. Instead of storing each of the 0 and 1 values, the system CPT is compressed using a combination of *runs* and *phrases* (Tien and Der Kiureghian 2016). A *run* is defined as consecutive repetitions of the same value. A *phrase* is defined as a repeated pattern in the values.

The system CPT is organized with $C_1, \ldots, C_n$ in the columns left to right as in Table 1 such that the value of C_n alternates in every row, the value of C_{n-1} alternates every 2 rows, and so on, until the value of C_1 alternates only once, taking the value of 0 for rows $1, \ldots, 2^{n-1}$, and the value of 1 for rows $2^{n-1}+1, \ldots, 2^n$. With this pattern, given a row number k, the state of component i, s_i, $i = 1, \ldots, n$, in that row of the CPT is 0 if $ceil(k/2^{n-i})$ is odd, and 1 if it is even. $ceil(x)$ is the value of x rounded up to the nearest integer. Thus, the authors find the states of all components in each row and the first n columns of the full CPT matrix shown in Table 1 need not be stored. Instead, the focus is on the 2^n-length rightmost column in the CPT, and the authors use the compression algorithm to reduce the elements to a combination of runs and phrases.

The compression algorithm operates row by row through the full CPT. In each row, once the states of the components are determined, they are checked against the set of minimum cut sets (MCSs) of the system to determine the state of the system. A MCS is a minimum set of components whose joint failure constitutes failure of the system. A system may have any number of MCSs, depending on the number of components and the system configuration. The value of the system state (0 or 1) is then compressed either as a run (as part of repeated instances of the same value) or a phrase (as part of a pattern that may repeat through the full length of the CPT, with phrases stored in a dictionary).

The compression algorithm results in a significantly reduced memory requirement for the BN model. For example, a sequence of 243 0's in the CPT is stored as a run of value 0 and length 243, i.e., as $\{\text{run}, 0, 243\}$, and the 600 element-length sequence $\{1, 0, 0, 1, 0, 0, \ldots, 1, 0, 0\}$ is reduced to the phrase $\{1, 0, 0\}$ repeated 200 times, i.e., to $\{\text{phrase}, 1, 200\}$ with the phrase labeled as Phrase 1 in the dictionary. These savings in memory, however, are accompanied by an increase in computation time (Tien and Der Kiureghian 2013). This is an example of the classic trade-off of storage space versus computation time, as described by Dechter (1999). In this case, the increase in computation time can be prohibitive in the modeling of large infrastructure systems.

The computational demand in the compression algorithm arises from the algorithm running through each of the 2^n distinct combinations of component states for each row of the CPT and comparing them with the set of MCSs. However, knowledge about the structure of the system can be used to reduce the number of rows to be analyzed. For example, if components C_1 and C_2 jointly constitute a MCS, i.e., joint failure of Components 1 and 2 leads to system failure, the authors need not check the states of other components when C_1 and C_2 are both in the failed state.

The first proposed heuristic in this paper uses this knowledge to reduce the computational demand by strategically specifying the order of the components. In general, determining the optimal ordering of nodes in a BN is a nondeterministic polynomial time (NP)-hard problem. For this heuristic, the authors first order the MCSs by size. The authors then order the components such that those in the smallest MCSs are numbered first and appear in the leftmost columns of the CPT. With this heuristic, the states of fewer components need to be checked against MCSs during compression. Furthermore, knowing where components in small MCSs appear in failed states in the CPT enables the authors to know which rows in the CPT need not be processed when running the compression algorithm. One can show that, given the states of the components $s_1, \ldots, s_n$ for a system of n components, the corresponding row number k in the CPT is given by

$$k = 1 + \sum_{i=1}^{n} s_i \times 2^{n-i} \tag{1}$$

For example, for a system of 10 components, the component states $\{1, 0, 0, 1, 0, 1, 1, 1, 1, 0\}$ appear in row $1 + 2^9 + 2^6 + 2^4 + 2^3 + 2^2 + 2 = 607$ of the CPT. Thus, given the failed states of components comprising a MCS, the authors can easily determine the rows where the system is in the failed state and no further matching with MCSs is necessary. These are rows where $s_i = 0$ for the components included in the MCS and $s_i = 0$ or 1 for the components not included in the MCS. For example, if component C_1 constitutes a MCS on its own, the authors know that for rows $1, \ldots, 2^{n-1}$ the system is in the failed state and no further analysis is needed. If, instead, components C_2 and C_3 constitute a MCS, then the authors know that for rows $1, \ldots, 2^{n-3}$ and $2^{n-1} + 1, \ldots, 2^{n-1} + 2^{n-3}$ the system is in the failed state. The authors call these *0 intervals*, i.e., intervals of rows in the system column of the CPT that are in the 0 state.

Fig. 2 shows the flowchart of the compression algorithm with this heuristic implemented. The inputs are the number of components n and the set of MCSs $\{MCS\}$. The outputs are the compressed system CPT, $cCPT_{sys}$, and the initial dictionary of phrases

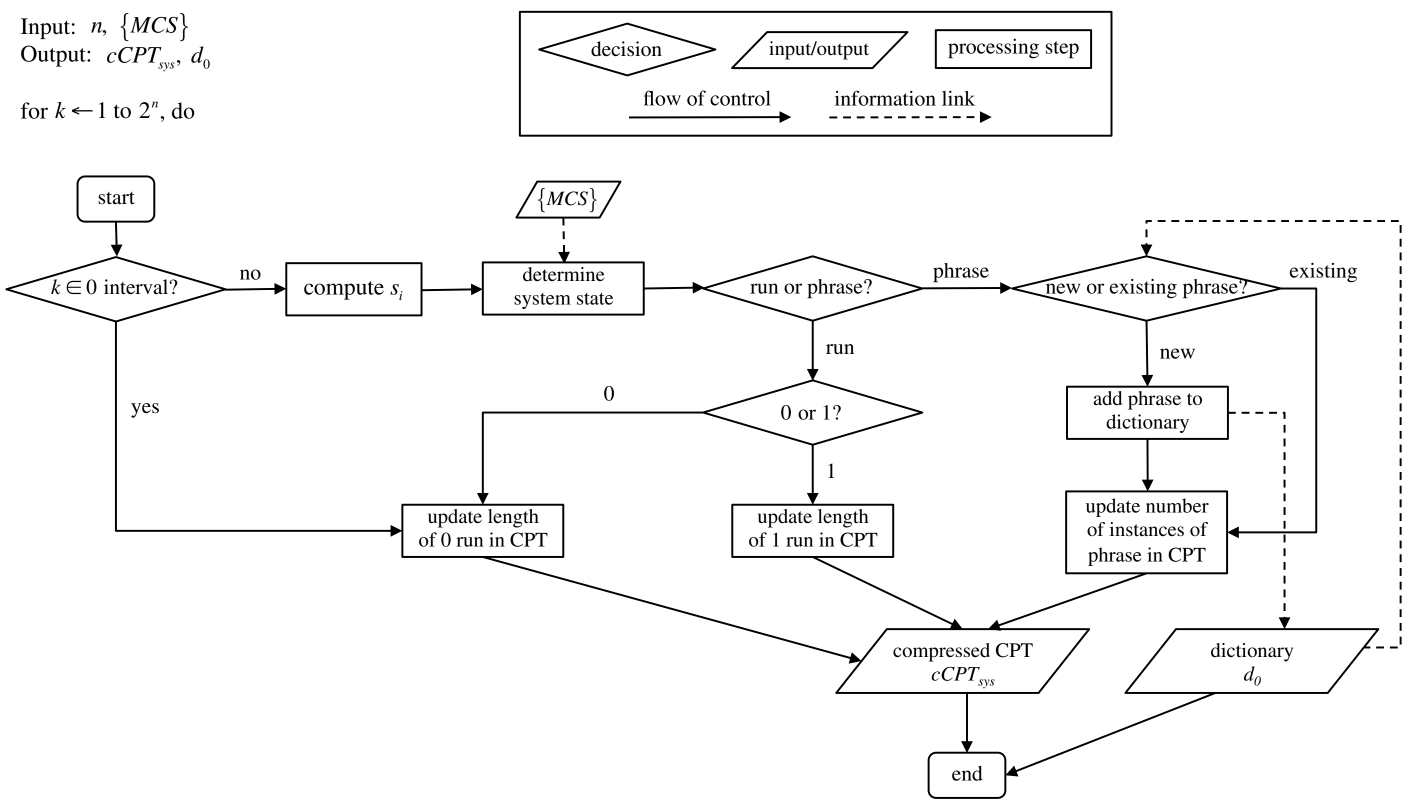

Fig. 2. Compression algorithm flowchart with component-order heuristic implemented

d_0. In the first step, if the row number k is determined to be within the 0 interval, then the authors need not continue through the bulk of the algorithm process. In a 0 interval, it is clear that the values are in a 0 run and the authors are able to directly move to updating the length of the 0 run in the CPT. Compared to previous studies [e.g., (Tien and Der Kiureghian 2016)], the authors need not compute the states s_i, check against $\{MCS\}$ to compute the system state, determine if the state is part of a run or a phrase, etc. Therefore, 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 for Computationally Efficient Inference

Once the BN has been constructed, Bayesian updating must be performed to draw inference about the system, e.g., posterior probabilities of the system or selected component states, given information about the state of certain components or the system itself. The inference algorithm previously developed by Tien and Der Kiureghian (2016) uses variable elimination (VE), whereby each node in the network is eliminated, one by one, until one arrives at a query node, a node for which the posterior probability is of interest. Elimination of node i results in the calculation of an intermediate factor, λ_i. The reader is referred to the work of Dechter (1999) for details on the VE algorithm.

To reduce the required memory storage, it is necessary for each intermediate factor λ_i to also be compressed. The developed inference algorithm handles both the system CPT and the intermediate factors λ_i in compressed form, without the need to decompress or recompress these matrices. Therefore, the memory storage savings achieved from the compression algorithm are preserved throughout the elimination process for inference. The computational time for inference, however, increases exponentially with size (Tien and Der Kiureghian 2013). Therefore, a methodology must be developed to increase computational efficiency of the inference algorithm and enable modeling of large infrastructure systems.

In this paper, the second heuristic and related algorithm address this need. Similar to construction of the BN, during VE, the selection of an optimal elimination ordering is an NP-hard problem. The heuristic for the inference algorithm aims at ordering the components in a particular way to improve computational efficiency, without solving the optimization problem. In the VE inference, all nodes other than the query node must be eliminated to arrive at the posterior probability distribution of the query node. During the node elimination process, when the authors 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 and processing through each row of λ_i to do so. This is a computationally demanding effort, particularly if the number of rows in λ_i is large.

The heuristic developed is to order the components such that query components appear as far to the left in the CPT as possible. This is similar to the compression algorithm heuristic. However, the greatest computational demand has been shown to be for compression (Tien and Der Kiureghian 2013). Therefore, combining this heuristic with that for the compression algorithm results in the components in the smallest MCSs being ordered first, directly followed by the query components. For the inference algorithm, this reduces the number of operations that must be performed to reorder λ_i when arriving at a query node during the VE process.

As an example, given a 10-component system, if C_9 is the query component, the inference algorithm requires processing through $2^9 = 512$ rows of λ_9 to reorder C_9 to the left and create the new λ_9. However, suppose C_1 and C_2 have been numbered first based on the compression algorithm heuristic. Next, the authors number the query component C_9 according to the inference algorithm heuristic. The inference algorithm is then able to run without any additional operations eliminating C_{10} and C_8 down to C_3, until it arrives at the query component C_9. In this step, reordering λ_3 is required. However, compared to the first case, the algorithm now needs to process through only $2^3 = 8$ rows of λ_3 for this step. 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, decreasing the time spent for inference. For each component that is eliminated, the size of λ_i decreases by a factor of 2. Therefore, in general, the number of operations is reduced by $2^{\hat{n}}$ for moving a query component $\hat{n}$ places to the left.

Algorithm for Supercomponents

The algorithms and heuristics presented thus far have been for a BN formulated as shown in Fig. 1, where each component is directly connected to the system node. In practice, infrastructure systems are often constructed with certain components in well-defined subsystems, e.g., with components arranged in series or parallel subsystems. In this case, systems can be more efficiently represented by grouping these subsets of components into supercomponents (SCs), as described by Pages and Gondran (1986) and used in the multiscale modeling approach described by Der Kiureghian and Song (2008) and Song and Ok (2010). In this section, the authors discuss the treatment of SCs in the developed algorithms.

For a system of n components, let $\{p_f\}$ denote the vector of length n defining the failure probabilities of components $C_1, \ldots, C_n$. Let n_{SC} denote the number of SCs in the system. Each SC comprises a simple subsystem of the overall system, specifically a subset of components that exist either in series or in parallel. The authors assume that each component of the system can be a member of at most one SC. Selection and definition of SCs is up to the modeler. Any given system can be divided into SCs in multiple ways. With series and parallel subsystems, the choice is clear, and the selection of SCs is done so as to be nonoverlapping. 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, $\{MCS\}$, is 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 subsystem of components, $cCPT_{SC}$ can be directly constructed without employing the compression algorithm described in Fig. 2. Let $\tilde{n}_i$ denote the number of components comprising the ith SC. Based on the definition of series and parallel systems, the $cCPT_{SC_i}$ for SC subsystems consists of two rows and is defined as follows:

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

where each row of the matrices corresponds to one row of the compressed SC CPT with three elements: whether the entry is a run or a phrase (run in all cases); the value of the run; and the length (number of repeated elements) of the run. As shown in Eq. (2), for the series subsystem, all elements of the CPT SC vector will be 0, except for the final row where all components in the SC are in the survival state and thus the SC is in the survival state. For the parallel subsystem, only for the first row, where all components in the SC are failed, will the SC be in the failed state. Otherwise, for all other rows in the CPT, the SC will be in the survival state. Since the $cCPT_{SC_i}$ is comprised only of runs, the accompanying phrase dictionary is $d_0 = 0$. For n_Q query nodes, let SC_Q denote the set of SCs that house the query nodes Q. Also, let $\{p_F|E\}$ denote the vector of length n_{SC} defining the conditional failure probabilities of the SCs, $SC_1, \ldots, SC_{n_{SC}}$, given the evidence E (the available information about the known states of components or system) so that its ith element is $\{p_F|E\}_i = \Pr(SC_i = 0|E)$, $i = 1, \ldots, n_{SC}$.

To perform inference on a system formulated in terms of SCs, the authors employ the following algorithm for supercomponents, which is a significant modification of the VE inference algorithm described by Tien and Der Kiureghian (2016). The compression algorithm referred to is the one shown in Fig. 2. In Step 4, in the construction of $cCPT_{SC_j}$, where $Q \in SC_j$ and $\tilde{n}_j > 1$, the authors need to *open* the SC_j to access the components inside the SC to perform inference on the query component(s). To do this, the authors use the output from Step 3, $\Pr(SC_Q|E)$, which gives the posterior state probabilities of the SCs with query nodes. Knowing the failure criteria for series and parallel subsystems, the authors can directly construct $cCPT_{SC_j}$ for series and parallel SCs as follows:

$$cCPT_{SC_j} = \begin{cases} \begin{bmatrix} \text{run} & \Pr(sys=1|SC_j=0) & 2^{\tilde{n}}-1 \\ \text{run} & \Pr(sys=1|SC_j=1) & 1 \end{bmatrix} & \text{for series SC} \\ \begin{bmatrix} \text{run} & \Pr(sys=1|SC_j=0) & 1 \\ \text{run} & \Pr(sys=1|SC_j=1) & 2^{\tilde{n}}-1 \end{bmatrix} & \text{for parallel SC} \end{cases} \qquad (3)$$

with the values in each row of the matrices corresponding to the same elements as in Eq. (2). With the compressed CPT, the authors are able to eliminate the nonquery components in SC_j. The result of the inference process as shown in Step 4 of the algorithm is the final output, the posterior probability of the specific query component Q of interest. An example implementation of this algorithm is given in the section describing the power distribution system application.

```
**Algorithm:** 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 Eq. (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 $\lambda_{k+1}$ 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 Eq. (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
```

Results for Test Application

The authors apply the proposed algorithms and heuristics to an example system, which is adopted from the eight-component system of Bensi et al. (2013). The performance of the system is measured by the ability to reach from source to sink through the labeled components $C_1, \ldots, C_n$. To analyze the performance of the algorithms in modeling systems of increasing size, the authors apply the algorithms to expanded versions of the example system, increasing the total number of components in the system to n. Fig. 3(a) shows the system obtained by increasing the number of components in the series subsystem. For this system, the set of MCSs is $\{MCS\} = \{(C_1, C_2, C_3, C_6), \ldots, (C_1, C_2, C_3, C_n), (C_4), (C_5)\}$ for a total of $n - 3$ MCSs. Fig. 3(b) shows the system obtained by increasing the number of components in the parallel subsystem to a total number of components n. For this system, the set of MCSs is $\{MCS\} = \{(C_1, \ldots, C_{n-5}, C_{n-4}), (C_1, \ldots, C_{n-5}, C_{n-3}), (C_1, \ldots, C_{n-5}, C_{n-2}), (C_{n-1}), (C_n)\}$ for a total of five MCSs. All calculations reported below were performed in *MATLAB* on a computer with 32 GB RAM and 2.2 GHz Intel Core i7 processor.

Results for Compression Algorithm

Fig. 4 shows the time required to compress the system CPT with (w/) and without (w/o) applying the compression algorithm heuristic to the expanded example systems. In the example systems, the two components to the extreme right of the system comprise a MCS on their own. Therefore, these components are numbered first and appear to the left in the CPT. Specifically, for the system in Fig. 3(a), components C_4 and C_5 are renumbered 1 and 2, and for the system in Fig. 3(b), components C_{n-1} and C_n are renumbered Components 1 and 2.

Fig. 4 shows significant reductions in computation times achieved by employing the heuristic for the compression algorithm. In addition, comparing the results for systems with increasing numbers of components in the series versus parallel subsystems, it is clear that the algorithm performs better for the latter case. Thus, the algorithm is better suited for systems formulated as a few MCSs of many components each, compared to systems formulated as many MCSs of few components each.

Results for Inference Algorithm

Fig. 5 shows the result of applying the inference algorithm heuristic to the expanded example systems. The computation times for a single run of the algorithms for forward (probability of system state given $C_1 = 0$) and backward (probability of component state given $sys = 0$) inference in systems with an increasing number of components in the series and parallel subsystems are plotted. The existing method for comparison is the widely used junction tree (JT) inference algorithm as implemented by Murphy (2001). Figs. 5(a–c) respectively show the results for the existing algorithm and the proposed new algorithm without and with the heuristic employed. When implementing the heuristic, the orders in which components appear in the CPT are $C_5, C_4, C_1, C_n, C_{n-1}, \ldots, C_2$ for the system in Fig. 3(a) and $C_n, C_{n-1}, C_1, C_{n-2}, C_{n-3}, \ldots, C_2$ for the system in Fig. 3(b).

Comparing Figs. 5(a and b), it can be seen 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 Fig. 5(c) it is clear that with

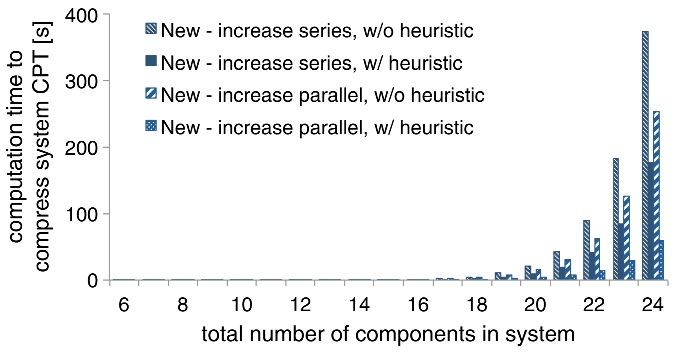

Fig. 4. Computation times to compress the system CPT without versus with heuristic implemented

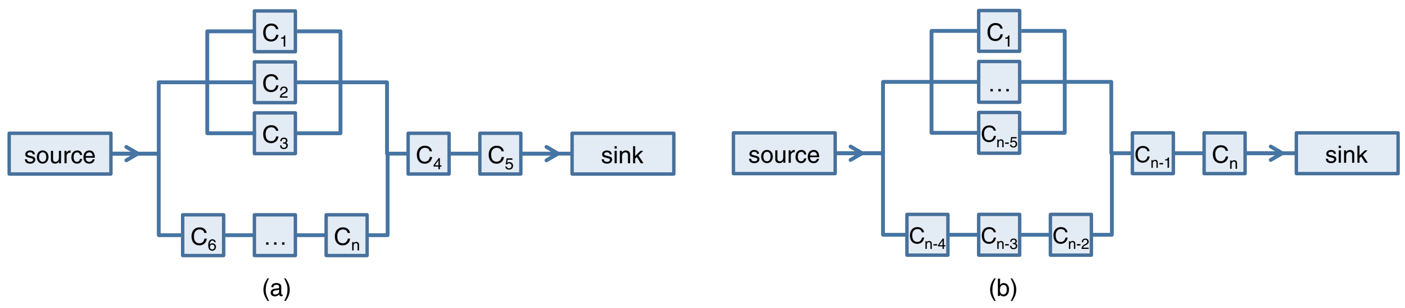

Fig. 3. Example test system: (a) expanded system with increased number of components in series subsystem; (b) expanded system with increased number of components in parallel subsystem

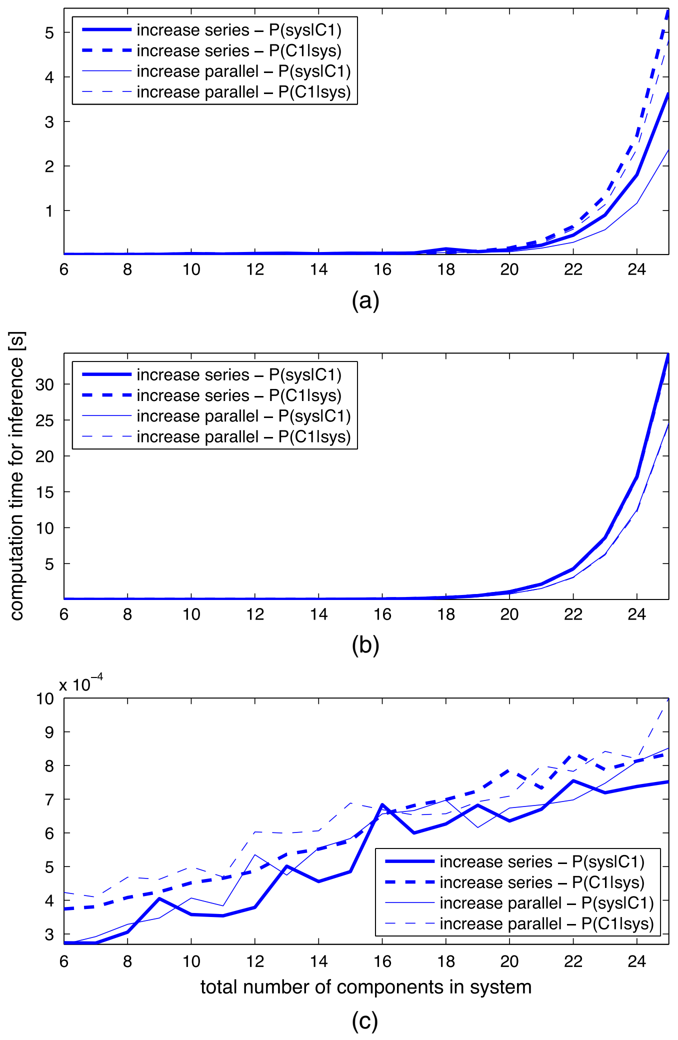

Fig. 5. Computation times for forward and backward inference using (a) the existing JT algorithm and the new inference algorithms; (b) without the heuristic implemented; (c) with the heuristic implemented

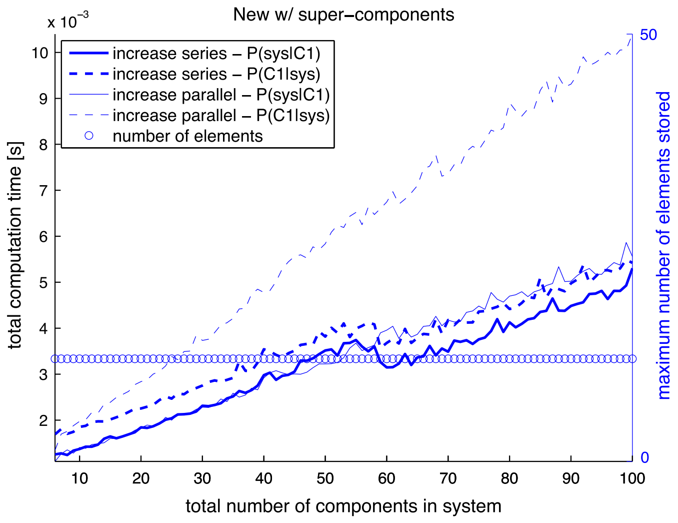

Fig. 6. Computation times (left ordinate) and memory storage (right ordinate) requirements for the new algorithm with supercomponents heuristic implemented

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 increase linearly, not exponentially, with system size. The reason for this is that when the heuristic is employed, the computation time becomes a function not of the full sizes of the intermediate factors λ_i, which exponentially increase with the system size, but of the size of the compressed λ_i's, which the authors showed previously to remain fairly constant with increasing system size (Tien and Der Kiureghian 2013, 2016). With the memory storage savings already demonstrated (Tien and Der Kiureghian 2016), these heuristics utilizing a more effective ordering of the components significantly improve the computational efficiency of both the compression and inference algorithms.

Results for Supercomponent Algorithms

Fig. 6 shows the result of applying the algorithms for SCs to the expanded example systems. Results are shown in terms of both computation time (left ordinate, solid and dashed lines) and memory storage (right ordinate, 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 is used as a proxy for the memory storage requirements of the new algorithm.

Fig. 6 shows that the total computation time is on the order of 10^{-2} s for a 100-component system. More importantly, the total computation time increases linearly with increasing system size. With regard to the memory storage, the maximum number of elements that must be stored remains constant, even as the total number of components in the system increases. Thus, at least for the example systems examined, the authors have achieved significant gains in both memory storage demand and computational efficiency with the new algorithms. These enable larger infrastructure systems to be modeled as BNs than is possible with previous methods.

Application to a Power Distribution System

With the objective of reliability assessment of more realistic infrastructure systems in mind, the authors now apply the algorithms and heuristics to the modeling and reliability assessment of the four-substation power network from Ostrom (2004) shown in Fig. 7, based on information from Pacific Gas and Electric, which was also investigated by Der Kiureghian and Song (2008). The power network consists of three inputs and one 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. The authors assume that the prior probability of failure of each component in the system is 0.1. Other fragility functions or models of component performance under different hazard scenarios can be integrated into the methodology through the component failure probabilities.

For this system, the authors implement the method of SCs, representing each triplet of switch-breaker-switch as a SC. Fig. 8 shows the resulting system, where the dashed squares represent SCs. It is this system of 59 components represented with 20 SCs that the authors use for the analysis.

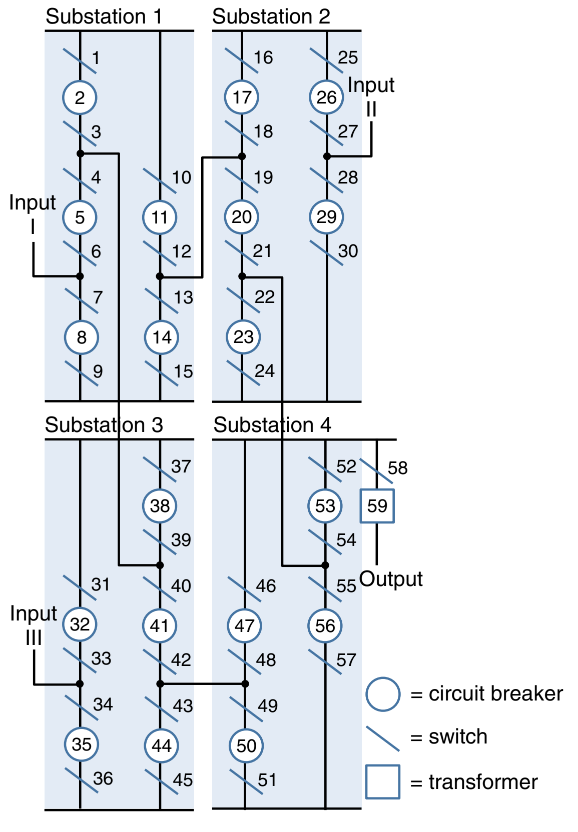

Fig. 7. Example power distribution system

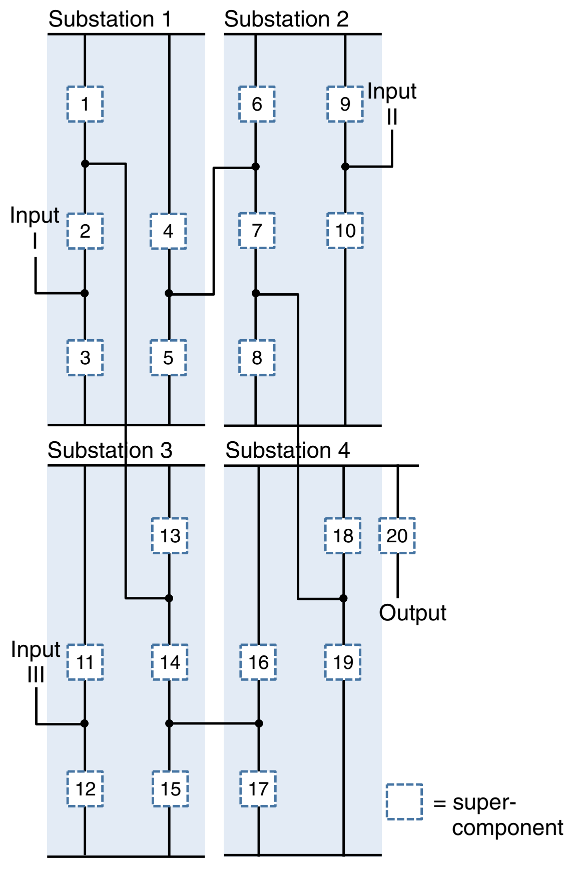

Fig. 8. Example power distribution system represented in terms of supercomponents

Implementation of Algorithms

The authors use this system to show an example implementation of the presented algorithms. Suppose the backward inference problem of obtaining the posterior distribution of C_1 given that the system has failed is of interest. C_1 is an element of SC_1. Using the algorithm for supercomponents, in Step 1, the authors define $cCPT_{SC_1}$ according to Eq. (2). As SC_1 is a series SC comprised of three components, the result is $cCPT_{SC_1} = \begin{bmatrix} \text{run} & 0 & 7 \\ \text{run} & 1 & 1 \end{bmatrix}$. Next, the authors eliminate the components in SC_1 for the given evidence. Using the VE inference algorithm with input $\hat{n}_1 = 3$ and query $Q = SC_1$, the output for the first element of $\{p_F|E\}$ is $\{p_F|E\}_1 = \Pr(SC_1 = 0|E) = 0.2710$. A similar process is performed for each SC in the system to obtain the full vector $\{p_F|E\}$.

In Step 2 of the algorithm, the compressed system CPT $cCPT_{sys}$ is constructed. For this example, $cCPT_{sys}$ is as shown in Table 2. No phrases are found, so the dictionary is $d_0 = 0$. $cCPT_{sys}$ is constructed using the compression algorithm and the set of minimum cut sets $\{MCS\}$ defined in terms of the SCs. The full $cCPT_{sys}$ consists of 176 rows, so it is shown in Table 2 in abbreviated form with an added column for repeated patterns of runs as indicated. In the algorithm, $cCPT_{sys}$ is stored in full form with 528 total elements. This is a lossless compressed representation of the $2^{20} = 1{,}048{,}576$ rows of the original CPT_{sys} (accounting for the SCs).

In Step 3, the posterior probability of C_1, which is a part of SC_1, is of interest. Thus, the set of SCs that house the query nodes is $SC_Q = SC_1$. To obtain the posterior state probability of SC_1, the authors perform VE with $Q = SC_1$ to eliminate all other SCs without query components until only SC_Q remains. This results in the output of the inference algorithm $\Pr(SC_Q|E) = 0.2729$.

Finally, in Step 4, the authors need to open SC_Q to perform inference on the query component within. For SC_1, which contains query component C_1, $\tilde{n}_1 = 3$, which is greater than 1. Thus, the authors construct $cCPT_{SC_1}$ according to Eq. (3). This results in the matrix $cCPT_{SC_1} = \begin{bmatrix} \text{run} & 0.7378 & 7 \\ \text{run} & 0.7403 & 1 \end{bmatrix}$. This is different from the original $cCPT_{SC_1}$ matrix as constructed using Eq. (2). This is because the authors are now beyond the step for initial construction of the CPTs, and $cCPT_{SC_1}$ can be thought of as an intermediate factor that gives the probability of survival of the overall system given the failure or survival states of the SC. VE is performed one final time with $cCPT_{SC_1}$ to eliminate the nonquery components C_3 and C_2, resulting in the final posterior probability for the query component, $\Pr(C_1|E) = 0.1007$.

Inference

The previous example illustrates the calculations to perform inference for one component. Given an infrastructure system comprised of many interconnected components, such as the example system in Fig. 7, system risk and reliability analyses enable identification of critical components to support decision making regarding inspection, repair, and replacement. These analyses are performed using inference. Figs. 9 and 10 show the results of performing forward and backward inference, respectively, for the example power system. By forward inference the authors mean determining the posterior probability of system failure given failure of a component. By backward inference, the authors mean determining the posterior failure probability of each component given failure of the system. Results are shown for component numbers in order of increasing posterior failure probabilities.

Fig. 9 gives the probability of system failure given component failure for each of the components 1–59. The results of this forward

Table 2. Compressed System CPT, $cCPT_{sys}$, for Power System Application

Run or phrase	r or p	L_r or n_p	Repeated pattern
Run	0	656,640	A
Run	1	256	
Run	0	128	
Run	1	2,432	
Run	0	1,024	
Run	1	3,072	
Run	0	1,280	
	$A \times 2$		
Run	1	2,816	B
Run	0	1,024	
Run	1	3,072	
Run	0	1,280	
	$B \times 4$		
Run	1	256	C
Run	0	128	
Run	1	2,432	
Run	0	1,024	
Run	1	11,264	
Run	0	128	D
Run	1	128	
	$D \times 1$		
Run	0	128	E
Run	1	384	
	$E \times 1$		
Run	0	128	F
Run	1	2,432	
	$D \times 1$		
Run	0	128	G
Run	1	11,904	
Run	0	1,280	
Run	1	2,816	
Run	0	1,024	
Run	1	27,648	
Run	0	8,192	
Run	1	8,192	
	$D \times 2,\ E \times 1,\ D \times 2,\ E \times 1,\ D \times 2,\ E \times 1,\ D \times 1$		
Run	0	128	H
Run	1	640	
	$D \times 1,\ H \times 1,\ D \times 1,\ H \times 1,\ D \times 1,\ H \times 1,\ D \times 1$		
Run	0	128	
Run	1	8,832	
Run	0	8,192	
Run	1	24,576	
Run	0	1,280	
	$C \times 1,\ D \times 2,\ E \times 2,\ F \times 1,\ D \times 1,\ G \times 1$		
Run	0	1,280	
	$C \times 1,\ D \times 2,\ E \times 2,\ F \times 1,\ D \times 1,\ G \times 1$		
Run	0	1,280	
	$C \times 1,\ D \times 2,\ E \times 2,\ F \times 1,\ D \times 1,\ G \times 1$		

inference support decision making in the management of the power system to minimize the risk of system failure. For example, knowing that the 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, it can be seen that, for example, Components 46–48 are more critical than Components 10–15. If each of these components is known to have

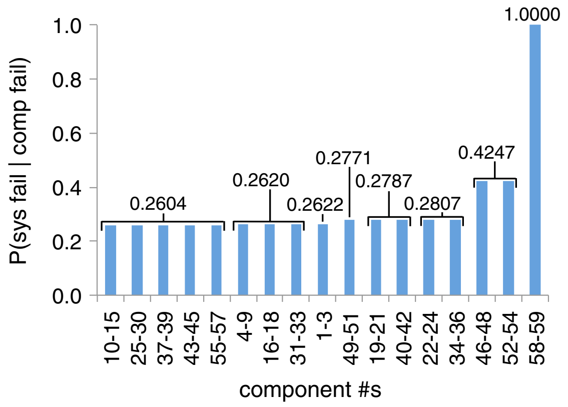

Fig. 9. Results of forward inference for power system application

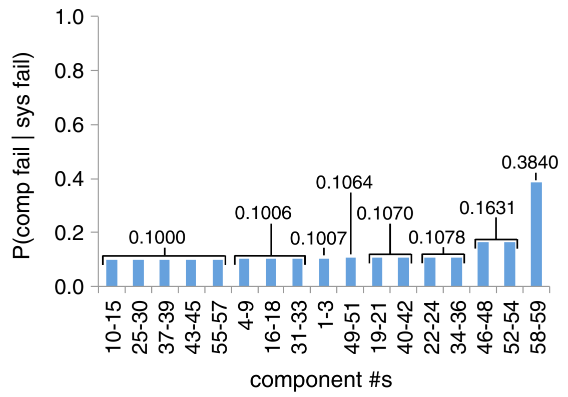

Fig. 10. Results of backward inference for power system application

failed, the updated probability of system failure is 0.4247 for the former set compared to 0.2604 for the latter set. Of course, results can also be generated for the posterior probability of the system failure for any set of multiple components known to have failed.

Fig. 10 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 Component 58 or 59 has 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. The evidence on the system state is not informative for these components, and the updated probability of failure virtually equals the prior failure probability.

These inference results enable the authors to identify the critical components of a system and inform decision making in the prioritization of limited resources in the management and rehabilitation of infrastructure systems.

Performance of Algorithms

For the power system shown in Fig. 7 and employing the compression algorithm heuristic and SC algorithm, the time required to compress the system CPT is 12.8 s. The computation times required for performing forward and backward inference, the results of which are shown in Figs. 9 and 10, are given in Figs. 11 and 12, respectively.

From Fig. 11, the average time required to compute the probability of system failure given component failure is 0.0212 s. In Fig. 12 for backward inference, there is an exponential increase

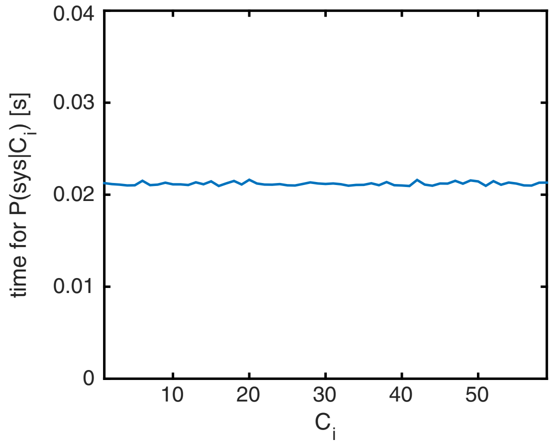

Fig. 11. Computation times for forward inference for power system application

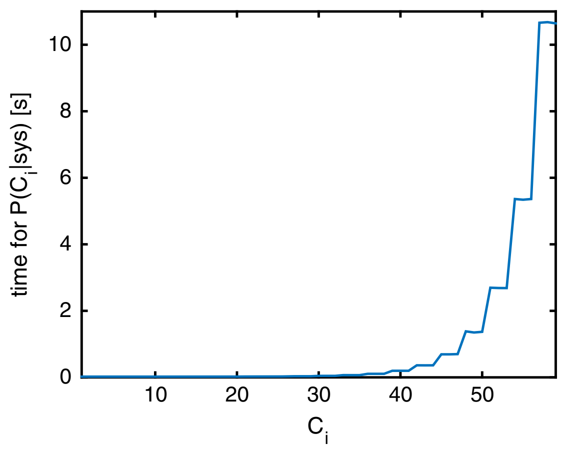

Fig. 12. Computation times for backward inference for power system application

in the computation time as the component number increases. This is because the backward inference time plotted for each C_i represents the result from the analysis with a fixed ordering of components. As described earlier, when arriving at a query node during the VE inference algorithm, the intermediate factor λ_i must be reordered, a computationally demanding operation. Therefore, query components appearing to the left in the CPT, i.e., with lower numbers i, require shorter computational times compared to those appearing to the right, i.e., with higher numbers i. Fig. 12 shows the need for the proposed inference algorithm heuristic, whereby query nodes are ordered as far to the left in the CPT as possible. When doing so, computation times for the query nodes will be similar to those for Components 1–20 in Fig. 12.

Finally, the authors look at the memory storage requirements. The maximum numbers of elements that must be stored to perform forward and backward inference are 707 and 1,059, respectively. These numbers should be compared with 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 new algorithms with only around 1,000 elements needed represents a many orders of magnitude reduction in memory demand. The authors conclude that the new algorithms together with the three heuristics achieve significant gains in both memory storage and computation time, enabling larger systems to be modeled as BNs for system reliability analysis.

Conclusion

In this paper, the authors presented methodologies to enable the BN modeling of critical infrastructure systems for reliability assessment. These include more computationally efficient algorithms for constructing the BN model and performing inference on the BN. Heuristics for the compression and inference algorithms utilize the ordering of the components to reduce the amount of needed computations. The third heuristic addresses the BN formulation of the system by employing the use of supercomponents. The heuristics and corresponding algorithms are applied to example systems, including a system of increasing size and a 59-component power distribution network, and the improvements in efficiency are demonstrated. With the heuristics employed, the new algorithms are shown to achieve significant gains in both memory storage and computation time, enabling the modeling of large infrastructure systems as BNs for system reliability analysis.

Acknowledgments

The first author acknowledges support from the National Science Foundation Graduate Research Fellowship from 2011 to 2014. Additional support was provided by the National Science Foundation Grant No. CMMI-1130061, which is also gratefully acknowledged.

References

Bensi, M., Der Kiureghian, A., and Straub, D. (2013). "Efficient Bayesian network modeling of systems." *Reliab. Eng. Syst. Saf.*, 112(3), 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, 337–349.

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

Der Kiureghian, A., and Song, J. (2008). "Multi-scale reliability analysis and updating of complex systems by use of linear programming." *Reliab. Eng. Syst. Saf.*, 93(2), 288–297.

Di Giorgio, A., and Liberati, F. (2011). "Interdependency modeling and analysis of critical infrastructures based on dynamic Bayesian networks." *Proc., 19th Mediterranean Conf. on Control and Automation*, IEEE, New York.

Di Giorgio, A., and Liberati, F. (2012). "A Bayesian network-based approach to the critical infrastructure interdependencies analysis." *IEEE Syst. J.*, 6(3), 510–519.

Jha, M. K. (2009). "Dynamic Bayesian network for predicting the likelihood of a terrorist attack at critical transportation infrastructure facilities." *J. Infrastruct. Syst.*, 10.1061/(ASCE)1076-0342(2009)15:1(31), 31–39.

Johansen, C., Horney, J., and Tien, I. (2017). "Metrics for evaluating and improving community resilience." *J. Infrastruct. Syst.*, 23(2), 04016032.

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

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.

Murphy, K. P. (2001). "The Bayes net toolbox for Matlab." *Comput. Sci. Stat.*, 33(2), 1024–1034.

Nielsen, T. D., Wuillemin, P. H., and Jensen, F. V. (2000). "Using ROBDDs for inference in Bayesian networks with troubleshooting as an example." *Proc., 16th Conf. in Uncertainty in Artificial Intelligence*, Stanford Univ., Stanford, CA, 426–435.

Ostrom, D. (2004). “Database of seismic parameters of equipment in substations.” ⟨http://peer.berkeley.edu/lifelines/lifelines_pre_2006/final_reports/413-FR.pdf⟩ (Mar. 1, 2014).

Pages, A., and Gondran, M. (1986). *System reliability: Evaluation and prediction in engineering*, Springer, New York.

Song, J., and Ok, S. Y. (2010). “Multi-scale system reliability analysis of lifeline networks under earthquake hazards.” *Earthquake Eng. Struct. Dyn.*, 39(3), 259–279.

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., Univ. of British Columbia, Vancouver, BC, Canada.

Tien, I., and Der Kiureghian, A. (2016). “Algorithms for Bayesian network modeling and reliability assessment of infrastructure systems.” *Reliab. Eng. Syst. Saf.*, 156(6), 134–147.

Tong, Y., and Tien, I. (2017). “Algorithms for Bayesian network modeling, inference, and reliability assessment for multi-state flow networks.” *J. Comput. Civil Eng.*, 10.1061/(ASCE)CP.1943-5487.0000699, 04017051.

Torres-Toledano, J. G., and Succar, L. E. (1998). “Bayesian networks for reliability analysis of complex systems.” *Ibero-American Conf. on Artificial Intelligence*, Springer, Heidelberg, 195–206.

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