12th International Conference on Applications of Statistics and Probability in Civil Engineering, ICASP12
Vancouver, Canada, July 12-15, 2015

Compression and Inference Algorithms for Bayesian Network Modeling of Infrastructure Systems

Iris Tien
Assistant Professor, School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, GA, USA

Armen Der Kiureghian
Professor, Department of Civil and Environmental Engineering, University of California, Berkeley, Berkeley, CA, USA

ABSTRACT: The Bayesian network (BN) is an ideal tool for modeling and assessing the reliability of civil infrastructure, particularly when the information about the system and its components is uncertain and evolves in time. One of the major limitations of the BN framework, however, is the size and complexity of the system that can be tractably modeled as a BN. This is due to the size of the conditional probability table (CPT) associated with the system node in the BN model, which grows exponentially with the number of components in the system. In this paper, we present novel compression and inference algorithms that utilize compression techniques to achieve significant savings in memory storage of the system CPT. In addition, heuristics developed to improve the computational efficiency of the algorithms are presented. An application to an example system demonstrates the gains in both memory and computation time requirements achieved by the proposed algorithms.

1. INTRODUCTION

Infrastructure systems are essential for a functioning society. Our nation's infrastructure is aging and becoming increasingly unreliable with potentially severe consequences. The Bayesian network (BN) is an ideal tool for modeling and assessing the reliability of civil infrastructure, particularly when information about the system and its components is uncertain and evolves in time. The major obstacle to the widespread use of BNs for system reliability analysis, however, is the limited size and complexity of the system that can be tractably modeled as a BN. This is due to the exponentially increasing number of elements that must be stored in the conditional probability table (CPT) for the system node in the BN as the number of components in the system increases.

In this paper, novel compression and inference algorithms developed to address this limitation are proposed. The algorithms are applied to the analysis of an example system, and the performance along both memory storage and computational efficiency metrics is evaluated compared to existing methods. The gains achieved by the developed algorithms enable larger systems to be modeled as BNs for system reliability analysis.

2. BACKGROUND AND RELATED WORK

Previous work using BNs for system reliability assessment have been limited to the study of small systems, e.g., systems comprised of 5, 8, and 10 components in Kim (2011), Mahadevan (2001), and Bobbio et al. (2001), respectively. Boudali and Dugan (2005) use BNs to model the reliability of slightly larger systems, including a system of 16 components. However, the authors state that this "large number" of components makes it "practically impossible" to solve the network without resorting to simplifying assumptions or approximations. It is clear that even a network of 16 components is not enough to create a full model of many real-world systems.

This limitation in system size is due to the conditional probability tables (CPTs) that must be associated with each node in the BN. The CPT defines the conditional probability mass function of the node, given each mutually exclusive combination of the states of its parent nodes. Consequently, the size of the CPT grows exponentially with the number of parents of the node. For example, if every node has m states, then the CPT has m^{n+1} entries, where n denotes the number of parents. For an infrastructure system comprised of n interconnected components $C_1, \ldots, C_n$, the state of each of the constituent components impacts the state of the overall system. Therefore, the BN is as shown in Figure 1.

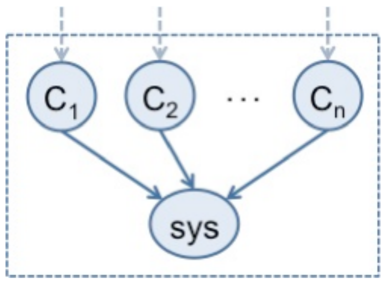

Figure 1: BN of system comprised of n components.

The individual component nodes are parents to the system node, denoted sys in the BN. Let us consider a binary system, where the components and the system are each in one of two possible states, e.g., survival or failure. For a system of 100 components, therefore, the CPT for the system node consists of $2^{101} = 2.5\times10^{30}$ individual terms. This poses a significant memory storage challenge for constructing and analyzing the BN.

One approach to address this limitation is to utilize Reduced Ordered Binary Decision Diagrams (ROBDDs), as in Nielsen et al. (2000), to efficiently perform inference in BNs representing large systems with binary nodes. However, a troubleshooting model is considered, which includes a major assumption of single-fault failures. It is this assumption that bounds the size of the ROBDD and enables modeling of larger systems. For more general systems, this single-fault assumption cannot be guaranteed, and the gains from using the ROBDD may not be applicable.

Another recent approach by Bensi et al. (2013) has been to optimize the topology of the BN to address the inefficiency of a converging BN structure as shown in Figure 1. This method seeks to create a chain-like BN model of the system with minimal clique sizes. The proposed optimization program, however, must consider the permutation of all component indices and, therefore, may itself become intractably large for large systems.

In this paper, we propose novel compression and inference algorithms to address the limitation in system size in BN modeling of infrastructure systems. Note that while the algorithm presented is for binary systems, it can be extended to multi-state flow systems, e.g., where the component states are discretized values of flow capacity, e.g., 0%, 50%, and 100% of maximum capacity. In addition, while the proposed algorithms are able to accommodate the case of dependent components, as indicated by the dotted arrows to parent nodes above the component nodes, this paper focuses on the system description part of the BN enclosed in the dashed box, which models the system performance.

3. PROPOSED ALGORITHMS

3.1. Compression Algorithm

For the BN representation of a system as shown in Figure 1, the component states deterministically define the system state. That is, for any combination of the component states, we know with certainty whether the system is in either a survival or a failure state. This model is particularly useful for studying the reliability of infrastructure systems, such as gas, power, or transportation systems, where the states of individual gas pipelines, electrical transmission lines, or roadways directly determine the state of the infrastructure system. When the component states deterministically define the system state, the CPT associated with the system node has a special property. For a binary system, let us define failure as 0 and survival as 1. Since for each distinct combination of component states the system state is known with certainty, the system CPT is comprised solely of 0s and 1s. The proposed compression algorithm takes advantage of this property. An example of the system CPT is shown in Table 1.

Table 1: Example conditional probability table for system node from BN of Figure 1.

C_1	$\cdots$	C_{n-1}	C_n	sys
0	$\cdots$	0	0	0
0	$\cdots$	0	1	0
0	$\cdots$	1	0	1
0	$\cdots$	1	1	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	$\cdots$	0	0	0
1	$\cdots$	0	1	1
1	$\cdots$	1	0	1
1	$\cdots$	1	1	1

Note that the right-most column gives the state of the system given each combination of states of the components. It is this vector, comprised solely of 0s and 1s, which we seek to compress. Specifically, the developed algorithm utilizes compression techniques, including run-length encoding and the classical Lempel-Ziv algorithm to compress the system CPT and achieve orders of magnitude savings in memory storage for the system CPT.

A run is defined as consecutive bits of the same value. Run-length encoding stores runs in a dataset as a data value and count, making it well suited for data with many repeated values. However, mixed values are stored literally, which results in little gain for mixed data. Algorithms based on the classical Lempel-Ziv algorithm (Ziv and Lempel 1977) find patterns in the data, construct a dictionary of phrases, and encode based on repeated instances of phrases in the dictionary. The proposed compression algorithm uses both these ideas to compress the system CPT of the BN as a combination of runs and phrases.

The proposed compression algorithm works by processing through the system CPT by row. Each row represents a distinct combination of component states. For each combination, a minimum cut set (MCS) or minimum link set (MLS) formulation of the system is used to determine the system state. A MCS is a minimum set of components whose joint failure constitutes failure of the system; if any one MCS fails, the system fails. Similarly, a MLS is a minimum set of components whose joint survival constitutes survival of the system; if any one MLS survives, the system survives. Each combination of component states in the system CPT (see Table 1) is checked against the MCSs or MLSs to determine the system state. Once the system state, i.e., 0 or 1, has been found, that value is encoded as a run or a phrase. The compression algorithm continues through the remaining rows of the system CPT until all rows have been processed. The resulting compressed combination of runs and phrases is typically orders of magnitude smaller than the size of the original CPT.

3.2. Inference Algorithm

Once the system CPT for the BN has been constructed in a compressed form, inference is required to draw conclusions about the system. We consider exact methods of inference since sampling-based approximate methods often yield poor convergence due to the small probabilities of events being modeled. Of the two major algorithms used for exact inference, the junction tree (JT) algorithm and the variable elimination (VE) algorithm (Dechter 1999), we use VE, as the BN in Figure 1 is comprised of only one clique of size $n + 1$, and computation of the potential over this clique increases exponentially as the size of the system increases.

In VE, nodes in the BN are eliminated, one by one, to arrive at the query node, the node for which the posterior (updated) probability distribution is of interest. Elimination of each node consists of summing the joint distribution over all states of the node, resulting in an intermediate factor λ that is used during the next step of elimination. For the system in Figure 1, suppose we are interested in the posterior distribution of the state of C_1, given a particular state sys of the system. The VE calculation for this query is

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

where CPT_{sys} is the compressed system CPT and λ_i is the intermediate factor created after the elimination of component C_i. The key to successfully performing inference for a large system is that each subsequent λ_i must also be compressed using the same compression algorithm as used for CPT_{sys}. The developed inference algorithm is able to perform VE on the compressed CPT_{sys} and λ_i matrices without decompressing or recompressing. Therefore, the memory storage gains achieved by compressing the system CPT are preserved throughout the process for inference.

4. RESULTS

4.1. Example System

We start with the 8-component example system shown in Figure 2, adopted from Bensi et al. (2013). The system consists of a parallel subsystem $\{C_1, C_2, C_3\}$ and series subsystems $\{C_4, C_5, C_6\}$ and $\{C_7, C_8\}$. Because the objective is to see how the proposed algorithm scales, we increase the number of components in the first two subsystems to a total number of components in the system n. The resulting analysis of these two expanded systems shows how the proposed algorithm performs compared to existing algorithms for systems of increasing size.

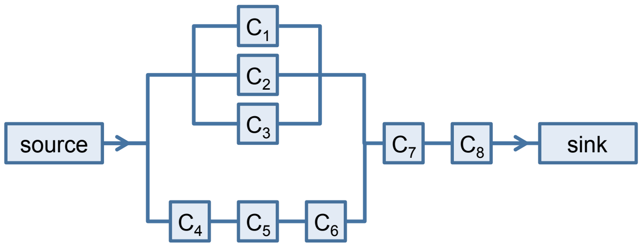

Figure 2. Example 8-component system, which is expanded to n components in following analysis.

4.2. Memory Storage

We examine the performance of the new algorithm compared to existing algorithms on two measures: memory storage and computation time.

Figure 3 shows the maximum number of values that must be stored in memory during the running of the algorithms, which is used as a proxy to assess the memory storage requirements of each algorithm. The algorithms are run in Matlab v7.10 on a 32 Gb RAM computer. Results from running the new algorithm are compared to using the existing JT algorithm, as implemented in the Bayes Net Toolbox by Murphy (2001).

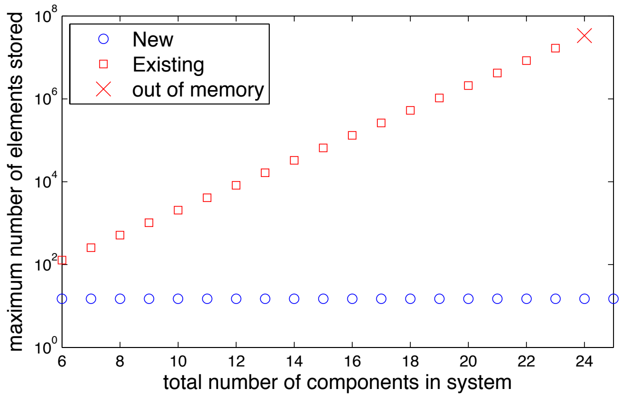

Figure 3. Number of values stored using the new vs. existing JT algorithm as a function of system size.

Figure 3 shows that the proposed algorithm achieves significant gains in memory storage demand compared to the existing algorithm. For the existing JT algorithm, the memory storage demand increases exponentially with the number of components in the system. In fact, the algorithm runs out of memory for a system comprised of more than 24 components. The memory storage demand of the proposed algorithm not only does not increase exponentially, but remains constant. The total number of values stored is 15, compared to 2^{n+1} for the size of the CPT.

While the size of this particular CPT, i.e., 15 elements, is specific to the system studied here, the orders of magnitude reduction in required memory storage is typical. The memory storage demand for a general system will depend on the number of runs and phrases in the CPTs and intermediate factors to be compressed, with the size of the compressed matrices a function of the number of switches between runs and phrases in the original matrices. Increasing the length of a given run or increasing the number of repeated instances of a given phrase, for example, will increase the number of rows in the original matrix being processed using the compression algorithm, but it will have no effect in increasing the memory storage demand of the proposed algorithm.

4.3. Computation Time

Figure 4 shows computation times required to run the new and existing algorithms with increasing system size. The computation times are broken into the various functions for each algorithm.

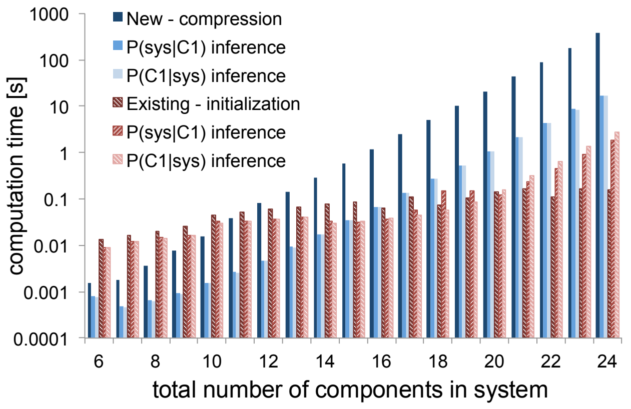

Figure 4. Computation times for the new vs. existing algorithm as a function of system size.

Taking Figures 3 and 4 together, we see the classic storage space versus computation time trade-off, as described in Dechter (1999). In Figure 4, we see that the gain in memory storage achieved by the new algorithm is accompanied by an increased computation time, as the new algorithm requires significantly longer computation time, especially during the initialization phase. However, the time to perform inference for both algorithms is exponentially increasing with the system size. One should note that, while analysis with the new algorithm may take longer, large systems simply cannot be solved using the existing algorithm due to exceeding the memory capacity.

4.4. Heuristic Augmentation

In this section, we present two heuristics to improve the computational efficiency of the proposed algorithm. The first aims to improve the efficiency of the compression process, while the second focuses on the efficiency of the inference calculations.

To compress the full system CPT, the compression algorithm must run through each of the 2^n distinct combinations of component states (rows of the CPT). This leads to an exponentially increasing computation time for compressing the system CPT. However, knowledge about the structure of the system can be used to reduce the number of rows to be analyzed. For example, if component C_i on its own constitutes a MCS, i.e., failure of C_i leads to system failure, we need not check the states of other components when C_i is in the failed state.

In general, determining the optimal ordering of nodes in a BN is an NP-hard problem. The heuristic employed here is to order the components in the system such that components that constitute MCSs on their own are numbered first and appear to the left in the CPT. Knowing where these components appear in the CPT enables us to know which rows in the CPT need not be processed when running the compression algorithm. Figure 5 shows the result of applying this heuristic.

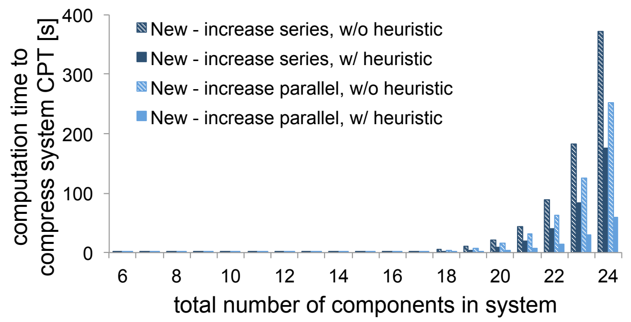

Figure 5. Computation times to compress the system CPT for systems with increasing number of components in the series vs. parallel subsystems, without vs. with the heuristic employed.

Figure 5 shows significant reduction in the computation times achieved by employing the above heuristic. In addition, we see that the algorithm performs better for systems with an increased number of parallel components compared to series components. The algorithm, therefore, is better suited to systems that have few MCSs comprised of many components each, compared to systems that have many MCSs of few components each. In the latter case, it would be preferable to use an MLS formulation of the system.

During the VE process for inference, all nodes other than the query node must be eliminated to arrive at the posterior probability distribution of the query node. When we arrive at the query node, it is necessary to move it to the left end of the CPT. This requires reordering of the elements in the λ_i factor, which is a computationally demanding effort. The heuristic employed here is to order the components such that query components appear as far to the left in the CPT as possible. This minimizes the number of operations that must be performed to reorder λ_i.

Figure 6 shows the result of applying this heuristic. The computation times for forward and backward inference in systems with an increasing number of components in the series and parallel subsystems are plotted. Figures 6(a), 6(b), and 6(c) respectively show the results from using the new algorithm without the heuristic employed, the existing JT algorithm, and the new algorithm with the heuristic employed.

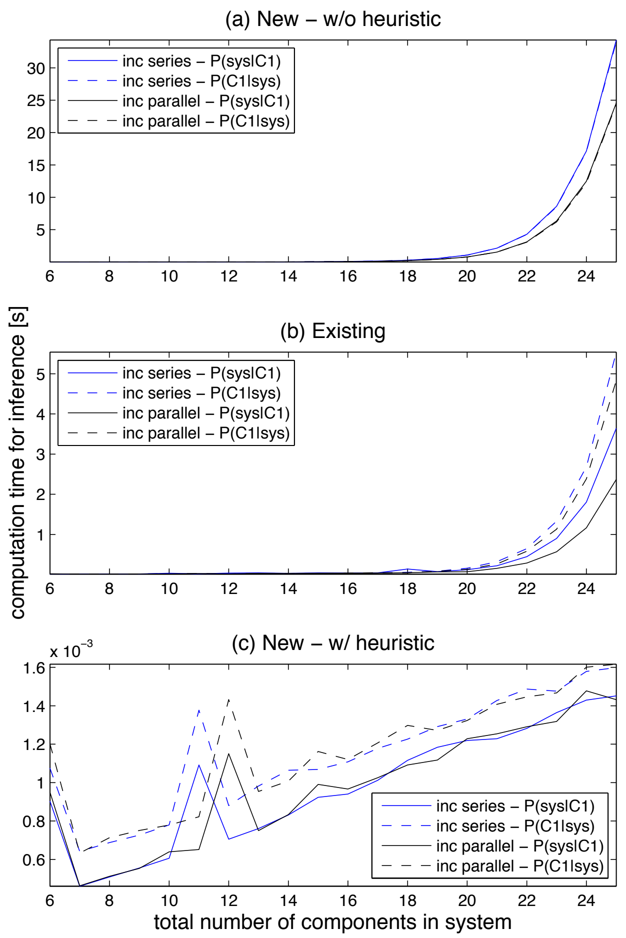

Figure 6. Computation times for forward and backward inference using the new vs. existing algorithm, without vs. with the heuristic employed.

Comparing Figures 6(a) and 6(b), we see that the new algorithm without the heuristic employed requires longer computation times for inference than the existing JT algorithm. In addition, the computation times for both algorithms increase exponentially as the system size increases. However, in Figure 6(c) we see that with the heuristic employed, the new algorithm achieves computation times that are orders of magnitude faster 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, the computation times are linearly, not exponentially, increasing with increasing system size. The reason for this is that when the heuristic is employed, the computation time becomes a function of not the full size of the λ_i's, which exponentially increase with the system size, but a function of the size of the compressed λ_i's, which we have seen remain constant with increasing system size. With the memory storage savings already demonstrated, these heuristics significantly improve the computational efficiency of the compression and inference algorithms, enabling large systems to be modeled as BNs.

5. CONCLUSION

We have developed a novel compression algorithm that achieves significant savings in memory storage of the system CPT for a BN model. In addition, we have developed an inference algorithm that operates on the compressed CPT's. It is shown that by implementing the proposed algorithms, the memory storage demand not only does not exponentially increase as the number of components in the system increases, but remains constant. We also present two heuristics, which significantly improve the computational efficiencies of the algorithms. Together, these algorithms enable large systems to be modeled as BNs for system reliability analysis.

6. REFERENCES

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

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

Boudali, H., and Dugan, J. B., "A discrete-time Bayesian network reliability modeling and analysis framework," *Reliability Engineering and System Safety*, Vol. 87, pp. 337-349, 2005.

Dechter, R., "Bucket Elimination: a Unifying Framework for Reasoning," *Artificial Intelligence*, Vol. 113, pp. 41-85, 1999.

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

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

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

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

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