Appeared in: *Reliability Engineering & System Safety*, 2013, **112**: 200–213.

Efficient Bayesian network modeling of systems

Michelle Bensi[1]*, Armen Der Kiureghian[2] and Daniel Straub[3]

[1] U.S. Nuclear Regulatory Commission, Washington, DC20555-0001, USA
[2] Department of Civil and Environmental Engineering, University of California, Berkeley, CA94720, USA
[3] Engineering Risk Analysis Group, Technische Universität München, Munich, Germany

ABSTRACT

The Bayesian network (BN) is a convenient tool for probabilistic modeling of system performance, particularly when it is of interest to update the reliability of the system or its components in light of observed information. In this paper, BN structures for modeling the performance of systems that are defined in terms of their minimum link or cut sets are investigated. Standard BN structures that define the system node as a child of its constituent components or its minimum link/cut sets lead to converging structures, which are computationally disadvantageous and could severely hamper application of the BN to real systems. A systematic approach to defining an alternative formulation is developed that creates chain-like BN structures that are orders of magnitude more efficient, particularly in terms of computational memory demand. The formulation uses an integer optimization algorithm to identify the most efficient BN structure. Example applications demonstrate the proposed methodology and quantify the gained computational advantage.

Keywords: Bayesian network, integer optimization, max-flow min-cut theorem, minimum cut sets, minimum link sets, parallel systems, series systems, systems.

* Formerly of University of California ,Berkeley, CA94720, USA.

1. Introduction

Engineering decisions often involve probabilistic assessment of the state of a system under evolving and uncertain information. For example, in the immediate aftermath of a natural disaster, such as an earthquake affecting an urban community, decisions must be made regarding dispatch of rescue teams and inspection crews, continued operation or closure of facilities, and prioritization of repair actions and restoration of services, all of which depend on the assessment of the functioning states of various infrastructural systems. Such assessment is strongly influenced by the available information, which in the immediate aftermath of a natural disaster is highly uncertain and rapidly evolves as the states of various system components are observed or measurements of the hazard are made. Another example is the management of a deteriorating system, where decisions need to be made on the frequency and extent of inspections and on maintenance, repair and replacement actions, while future capacities and demands of the system remain uncertain. In both cases, there is need for a method to update the probabilistic assessment of the system state as information, often of uncertain type, becomes available from measurements, inspections, and other observations of the system and its components.

The Bayesian network (BN) is an ideal framework for the analysis of such systems, particularly when updating of the probabilistic model in light of evolving and uncertain information is an important objective. The BN is a graphical model consisting of nodes and directed links, which respectively represent random variables and their probabilistic dependencies (Pearl 1988, Jensen and Nielsen 2007). The variables may represent the states of the components of a system, or their capacities and demands. The BN provides a convenient means for modeling dependence between the component states, which is rather difficult in most classical system reliability methods (Pagès and Gondran 1986). Furthermore, upon entering evidence on one or more variables, e.g., the observed states, capacities or demands of a subset of the components, the information propagates throughout the network and updates distributions of other random variables, e.g., the states of other components and the system, in accordance with the Bayes' rule. Finally, by addition of decision and utility nodes, the BN renders a decision graph that facilitates decision-making in accordance with the maximum expected utility criterion (Shachter 1986, Jensen and Nielsen 2007).

This paper focuses on the development of a systematic approach to using BNs for modeling the performance of systems that are defined in terms of their minimum link sets (MLSs) or minimum cut sets (MCSs). The methodology presented in this paper is motivated by our efforts in modeling the performance of spatially distributed civil infrastructure systems (e.g., a highway network or water distribution system) subjected to an earthquake hazard, with particular emphasis on post-earthquake risk assessment and decision making (see Bensi et al. (2011)). For such an application, efficient computations and near real-time inference in models of large systems are essential. Furthermore, the considered systems are more easily characterized in terms of their MLSs/MCSs than by other means, such as fault trees, event trees or reliability block diagrams. While this paper was motivated by this specific application, the methods presented are applicable to a broader scope of problems.

The BN has been used in the past for system reliability analysis (see, e.g., Torres-Toledano and Sucar 1998; Mahadevan et al. 2001; Bobbio et al. 2001; Friis-Hansen 2004; Liu et al. 2008; Lampis and Andrews 2009; Straub and Der Kiureghian 2010a-b). Some of these works consider intuitive approaches to modeling systems as BNs (e.g., Friis-Hansen 2004). Others consider the fault tree (e.g., Liu et al. 2008; Bobbio et al. 2001) or the reliability block

diagram (e.g., Torres-Toledano and Sucar 1998) as the native source of system information. Other papers develop BNs for relatively simple systems, for which computational demands are not of particular concern. This work differs from previous efforts by defining a systematic approach to developing an efficient BN structure for modeling the reliability of complex systems when the MLSs or MCSs are the native source of system information. The approach is particularly useful when working with topologically defined systems, in which the system decomposition is commonly done through the MLSs and/or MCSs. While other authors have used BNs to model systems that are topologically defined through a reliability block diagram (e.g., Torres-Toledano and Sucar 1998)), no systematic attempt has been made to optimize the BN structure, particularly when working with large, multi-state systems. It turns out that conventional BN models rapidly grow in size and density with increasing size of the system, so that even for moderately sized systems the computational and memory demands make the model infeasible, especially when using exact inference algorithms with multi-state nodes. With this shortcoming in mind, in this paper we develop methods for generating efficient BN topologies for modeling systems with binary and multi-state components. A discrete optimization algorithm is developed that minimizes the density of the BN, thereby providing orders of magnitude savings in computational time and memory. This development facilitates consideration of systems, which otherwise could not be solved with conventional BN formulations.

The paper begins with a brief introduction to the BN. The introduction is limited to those aspects that are needed to motivate the remainder of the paper. Next, efficient Bayesian network formulations for modeling series and parallel systems with binary components are presented. These are then extended to general systems with binary and multi-state components. To automate construction of the efficient Bayesian network formulations, a binary integer optimization problem is formulated. Furthermore, two heuristic augmentations are presented to reduce the size of the optimization problem. Several examples demonstrate the proposed methodology and its effectiveness.

2. Brief introduction to Bayesian networks

A BN is characterized by a directed acyclic graph consisting of a set of nodes representing random variables and a set of links representing probabilistic dependencies. In this paper, we limit the treatment to BNs in which all random variables are discrete; the reader interested in BNs with continuous random variables is referred to Langseth et al. (2009). Consider the simple BN in Figure 1. The directed links from X_1 and X_2 to X_3 indicate that the distribution of X_3 is defined conditioned on X_1 and X_2. In the BN terminology, random variable X_3 is said to be a *child* of random variables X_1 and X_2, while the latter are the *parents* of X_3. Similarly, X_4 is a child of X_1, while X_5 is a child of X_4. Each node is associated with a set of mutually exclusive and collectively exhaustive states, corresponding to the outcome space of the discrete random variable. Attached to each node is a *conditional probability table* (CPT), providing the conditional probability mass function of the random variable given each of the mutually exclusive states of its parents. For *root* nodes that have no parents, e.g., X_1 and X_2 in Figure 1, a *marginal probability table* is assigned.

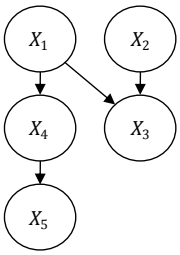


Figure 1. A simple BN

The joint distribution of the random variables in the BN is given as the product of the conditional distributions, i.e.,

$$p(x_1, x_2, \dots, x_n) = \prod_{i=1}^{n} p(x_i|\mathrm{Pa}(x_i)) \tag{1}$$

where $\mathrm{Pa}(x_i)$ is the set of parents of node X_i, $p(x_i|\mathrm{Pa}(x_i))$ is the CPT of X_i and n is the number of random variables (nodes) in the BN. Thus, for the BN in Figure 1, the joint probability mass function is

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5|x_4)p(x_4|x_1)p(x_3|x_1, x_2)p(x_1)p(x_2) \tag{2}$$

BNs are useful for answering probabilistic queries when one or more variables are observed. As an example, suppose for the BN in Figure 1 the observations $X_3 = x_3$ and $X_4 = x_4$ have been made and the conditional distribution $p(x_2|x_3, x_4)$ is of interest. This posterior distribution is computed by first marginalizing the joint distribution in (2) to obtain the joint distributions of the subsets of the variables:

$$p(x_2, x_3, x_4) = \sum_{x_1, x_5} p(x_1, \dots, x_5) \tag{3}$$

$$p(x_3, x_4) = \sum_{x_1, x_2, x_5} p(x_1, \dots, x_5) \tag{4}$$

The desired conditional distribution then is $p(x_2|x_3, x_4) = p(x_2, x_3, x_4)/p(x_3, x_4)$. While it is possible to obtain updated distributions by this method, this is not a computationally efficient approach for non-trivial BNs. Several efficient algorithms for exact and approximate probabilistic inference in BNs have been developed (see, e.g., Dechter 1996; Langseth et al. 2009; Lauritzen and Spiegelhalter 1988; Madsen 2008; Yuan & Druzdzel 2003; Yuan & Druzdzel 2006). The general principles of exact inference algorithms are outlined here to highlight the requirements for efficient BN topologies.

The efficiency of the BN stems from the decomposition of the joint distribution into local conditional distributions, as exemplified in Eq. (2). When summing over the joint distribution, as in Eqs. (3) and (4), no use of the decomposition is made and computations are inefficient. However, by writing the joint distribution in the product form of Eq. (1), it is possible to rearrange the summation and product operations due to their distributive and commutative properties. As an example, Eq. (3) is written as

$$p(x_2, x_3, x_4) = \sum_{x_1} \sum_{x_5} p(x_5|x_4)p(x_4|x_1)p(x_3|x_1, x_2)p(x_1)p(x_2) \tag{5}$$

$$= p(x_2) \sum_{x_1} p(x_4|x_1) p(x_3|x_1, x_2) p(x_1) \sum_{x_5} p(x_5|x_4)$$

The summation operations can be interpreted as node eliminations. Since calculations are performed from right to left, Eq. (5) corresponds to an elimination of X_5 followed by the elimination of X_1. Clearly, solving the second line of Eq. (5) is more efficient than solving the first, because the summations are performed in smaller domains. The summation over X_5 is in the domain of X_4 and X_5 only (and can actually be omitted, since it results in 1). The summation over X_1 is in the domain of X_1, X_2, X_3 and X_4, for which it is required to establish a table (called a *potential*), whose number of entries is equal to the product of the number of states of these variables. In general, the sizes of the potentials over which summations are performed determine the efficiency of the inference algorithm.

The potentials, and consequently the efficiency of the inference algorithm, depend on the ordering of node eliminations. However, there exist computationally optimal elimination sequences, all of which lead to the same domain set, i.e. the domains of the potentials arising in the process are the same. The domains corresponding to the optimal elimination sequences are called *cliques*. The total size of the potentials associated with these cliques is a good proxy for the computational effort required for performing inference in the BN and is used in this paper to assess and compare efficiencies of various BN system formulations.

While all exact inference algorithms aim at finding the optimal ordering of node eliminations, they follow different strategies for doing so. In particular, some algorithms optimize the elimination for a specific inference task, while others, such as the junction tree algorithm (Jensen & Nielsen 2007), optimize computations for general inference. With the latter, parts of the computations are reused, which becomes efficient when the interest is in updated distributions of many variables and when considering multiple evidence cases. Our interest is more focused on general inference applications. Several of these algorithms are implemented in available software applications (e.g. DSL 2007; Hugin Expert A/S 2008), facilitating inference in complex BNs.

While this paper focuses on improving the computational efficiency of exact inference algorithms, it is believed that the approaches developed here will also be useful for approximate algorithms by reducing the size of CPTs that must be stored. We do not explore computational improvements that might be available for exact inference by consideration of deterministic substructures within the BN. As Nielsen et al. (2000) have shown, advanced Boolean calculations can be used to improve the efficiency of exact algorithms when the deterministic part of a BN model is represented as a Boolean function. It is believed that such an approach can be used to further improve the computational efficiency of the BN model resulting from our topology optimization scheme.

3. Modeling system performance via Bayesian network

Consider a system of N_c components, with component i having n_i discrete states. The number of distinct configurations of the system is $\prod_{i=1}^{N_c} n_i$. We refer to a BN formulation that defines the system state directly in terms of the states of its constituent components as the *naïve BN formulation.* Figure 2 shows one such BN, where node C_i defines the state of component i and node S_{sys} defines the state of the system. If the system has N_s discrete states, then the CPT associated with the system node has $N_s \times \prod_{i=1}^{N_c} n_i$ entries. Although this

formulation has been used in the past (e.g., see Mahadevan et al. 2001), for a system with even a moderate number of components or component states, the size of the CPT for the system node becomes so large that the BN becomes computationally intractable. Clearly, a more efficient BN formulation is needed.

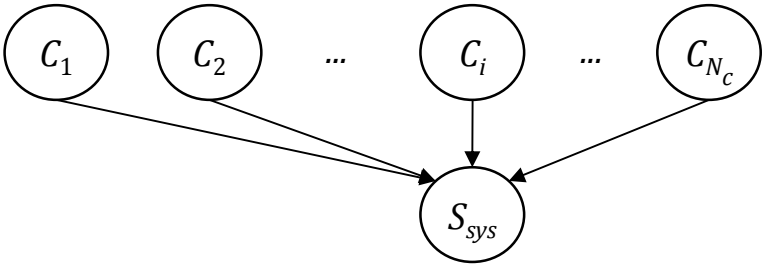


Figure 2: Naïve BN formulation

As an alternative to the above naïve approach, in this paper we present four additional BN formulations for modeling system performance. The first two formulations make use of minimal link sets (MLSs) and minimal cut sets (MCSs) for systems with binary component and system states, but employ converging structures as in the naïve formulation. It is shown that the MCS formulation is also applicable to a certain class of systems with multi-state components. We then develop two additional formulations that employ MLSs/MCSs for systems with binary states, but result in chain-like BN topologies that are far more efficient for computational purposes. These are first developed for series and parallel systems and then extended to general systems. The efficient MCS formulation is shown to be also applicable to a certain class of systems with multi-state components.

4. BN system models using minimal link and cut sets

4. 1 Binary components and system

Consider a system with binary component and system states, say survival/fail states. A minimal link set (MLS) is a minimum set of components whose joint survival constitutes survival of the system. The *minimal link set BN formulation* introduces intermediate nodes between the component and system nodes, which represent the states of the MLSs, as shown in Figure 3. Torres-Toledano and Sucar (1998) used such a BN formulation for modeling system performance, though with less formality and generality than described here. The binary states of the MLS nodes are defined such that each MLS node is in the survival state only if all its constituent components have survived; otherwise, it is in the fail state. The system node is in the survival state if any MLS node is in the survival state. Let N_{MLS} denote the number of MLSs of the system and $N_{MLS,i}$ denote the number of components in the ith MLS. The size of the CPT of the ith MLS node is $2^{N_{MLS,i}+1}$, and the size of the CPT of the system node is $2^{N_{MLS}+1}$. Clearly, when the number of MLSs is large, the size of the CPT associated with the system node becomes large. A similar problem occurs for an MLS node when the number of its constituent components is large.

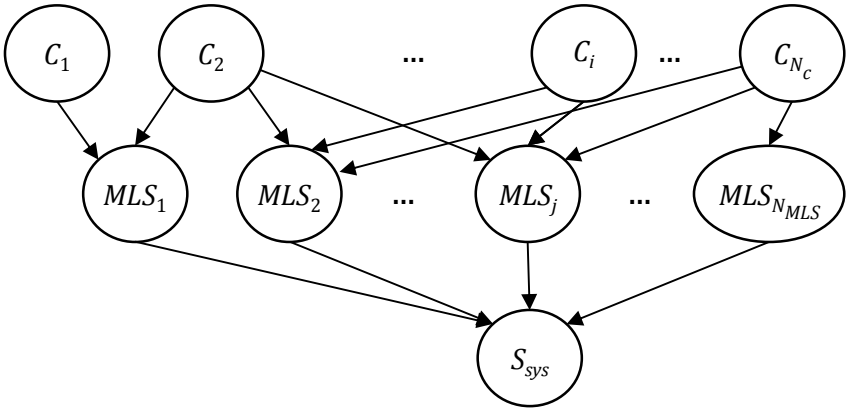


Figure 3: MLS formulation

The dual of the MLS formulation is the *minimal cut set BN formulation*. A MCS is a minimum set of components whose joint failure constitutes failure of the system. In this formulation, the system node is a child of nodes representing MCSs, and each MCS node is a child of nodes representing the states of its constituent components, see Figure 4. The system node is a series system of all the MCS nodes, (i.e., the system node is in the fail state if at least one MCS node is in the fail state), whereas each MCS is a parallel system of its parent nodes (i.e. all constituent components must be in the fail state for the MCS node to be in the fail state). As with the MLS formulation, the CPTs in this formulation become large as the number of MCSs, denoted N_{MCS}, increases and/or the number of components in an MCS, denoted $N_{MCS,i}$ for the ith MCS, becomes large.

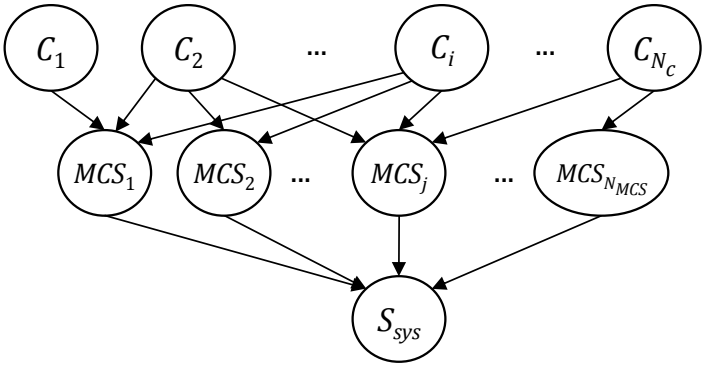


Figure 4: MCS formulation

4.2 Multi-state flow systems with multi-state components

A multi-state flow system with multi-state components can be modeled by the MCS BN formulation through application of the Max-Flow Min-Cut theorem (Elias et al. 1956; Ford and Fulkerson 1956). We are not aware of a similar theory that allows adaptation of the MLS formulation to multi-state flow systems.

Consider a system describing flow from a source node to a sink node and identify its MCSs assuming components are binary. Each component of the system has a flow capacity, which is discretized into a set of distinct states, e.g., 0%, 25%, 50%, 75% and 100% of a maximum capacity. Let Cap_i denote the capacity state of component i. For a distributed component

with directed flow, we consider the capacity when going from the source side to the sink side of the cut. Assign to each MCS a capacity value equal to the sum of the capacities of its constituent components. The Max-Flow Min-Cut theorem states that the maximum flow capacity from the source to the sink is equal to the minimum value among all MCSs, i.e. the bottleneck in the system. The theorem allows adaptation of the MCS BN formulation to multi-state problems, without changing the topology of the BN created when assuming components are binary. It is only necessary to increase the number of states associated with each BN node to correspond to the component, MCS or system capacity levels, and use arithmetic expressions rather than Boolean logic to define the relationships between the nodes, as described below.

Nodes C_i in Figure 4 are modified to represent multiple states corresponding to the capacity levels of each component. If the component has a continuous capacity state, then the range of possible capacities must be discretized into several small intervals; in such a case, node C_i is said to represent an interval node. Similarly, nodes MCS_i have multiple states having capacity values defined using the relation

$$Cap_{MCS_i} = \sum_{C_j \in MCS_i} Cap_j \tag{6}$$

The capacity of the system node is obtained using the relation

$$Cap_{sys} = \min_{i \in [1,\ldots,N_{MCS}]} Cap_{MCS_i} \tag{7}$$

When C_i are interval nodes, MCS_i and S_{sys} are also interval nodes. In constructing the CPTs of the latter nodes, the intervals for these nodes must be selected by considering the entire range of their possible capacity values as obtained from Eqs. (6) and (7). More details about computation of CPTs for interval nodes is given in Bensi et al. (2011).

5. Efficient BN system models

5.1 Motivation

The MLS and MCS formulations described in the preceding section result in converging BN structures. In general, BN structures with nodes arranged in chains are significantly more efficient than those having converging structures. As we will show later, the two BN structures shown in Figure 5 represent the same system. Figure 5a shows a converging structure similar to the BN models presented in the previous section, and Figure 5b illustrates a chain structure. In both BNs, dependence among the components is considered by introducing a common demand node D. A formal description of the construction of the BN in Figure 5b is presented later in this section.

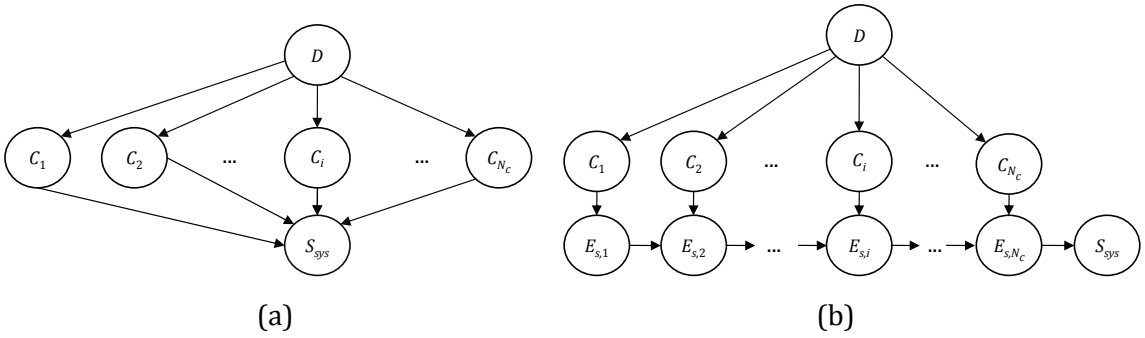


Figure 5: Equivalent BNs with (a) converging structure, and (b) chain structure

Figure 6 compares the computational demands, measured in terms of the total clique sizes, for the BNs in Figure 5 with converging and chain structures, when the system components have 2 and 5 states. As the number of components increase, the computational demand of the BN with converging structure increases exponentially, while that of the BN with the chain structure increases linearly. However, for 2-state components, the converging structure is more efficient than the chain structure when the number of components is less than 4. Thus, when the node modeling system performance in Figure 5a has more than 3 parents (i.e. constituent components), it is advantageous to model the system as in Figure 5b.

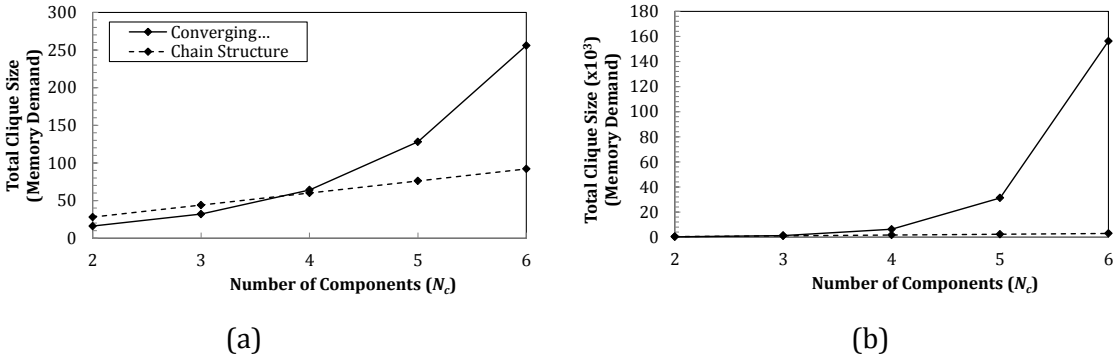


Figure 6: Computational demands of system BNs with converging and chain topologies when components have (a) 2 states and (b) 5 states

The computational demands associated with inference are influenced not only by the system size and configuration, but also by the number of common parents to the component nodes (similar to node D in Figure 5). Figure 7 shows a comparison of computational demands associated with the converging and chain BN topologies with binary component nodes with 1, 2 and 3 common parent demand nodes. It is observed that increasing the number of common parent nodes increases the computational demand. However, the chain structure remains advantageous to the converging structure as the number of components increases. Note that, as the number of common demand nodes increases, the "cut-off" point at which the chain structure is more efficient than the converging structure moves slightly upward. For three common demand nodes, the chain structure is more efficient for 5 or more components and for one common demand node, it is more efficient for 4 or more components.

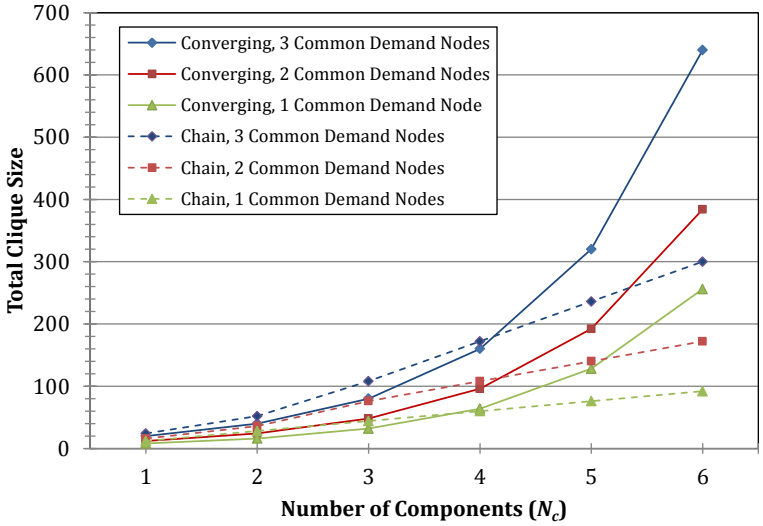


Figure 7: Comparison of computational demands associated with converging and chain BN topologies for binary nodes with one or more common parent demand nodes

5.2 Series and parallel systems with binary components

We now describe how the performance of systems with binary components can be modeled with BNs having the chain topology. Define a *survival path sequence* (SPS) as a chain of events, corresponding to a MLS, in which the terminal event in the sequence indicates whether or not all the components in the MLS are in the survival state. Note that the term "sequence" does not have any temporal implications. A series system has one MLS and a parallel system has N_c MLSs. It follows that a series system has one SPS and a parallel system has N_c SPSs. A SPS is comprised of a chain of *survival path events* (SPEs), each of which is associated with a component and describes the state of the sequence up to that event. SPEs are represented in the BN by nodes labeled $E_{s,i}$, the subscript i indicating the association with component i. The state of $E_{s,i}$ is defined as

$$\begin{aligned} E_{s,i} &= 1 \quad \text{if} \left\{ E_{s,\text{Pa}(i)} = 1 \right\} \cap \{C_i = 1\} \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{8}$$

where $E_{s,\text{Pa}(i)}$ defines the state of the SPE node that is parent to $E_{s,i}$; $E_{s,i} = 1$ indicates that the node is in the survival state and $E_{s,i} = 0$ indicates its failure (we use this Boolean notation throughout this paper). Thus, for a series system, the BN formulation takes the form shown in Figure 8a. The state of node $E_{s,1}$ is equal to the state of node C_1. $E_{s,2}$ is in the survival state only if $E_{s,1}$ is in the survival state and C_2 is in the survival state. This pattern continues such that E_{s,N_c} is in the survival state only if both E_{s,N_c-1} and C_{N_c} are in the survival state. Consequently, the state of E_{s,N_c} describes the state of the entire SPS (i.e. it indicates whether all components in the MLS have survived) and, therefore, that of the series system. The state of node S_{sys} is equal to the state of E_{s,N_c} in Figure 8a.

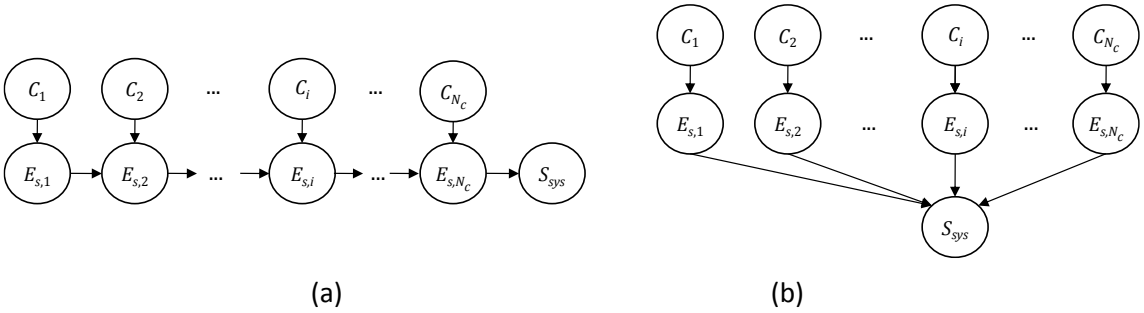


Figure 8: BN using SPEs to model performance of (a) series system, (b) parallel system

A parallel system has a SPS corresponding to each component. The resulting BN formulation is shown in Figure 8b. The system node indicates system survival if any node $E_{s,i}$ is in the survival state. Like the naïve formulation, the exponential growth in the size of the CPT associated with the system node renders this BN intractable when the number of components is large.

Define a *failure path sequence* (FPS) as a chain of events, corresponding to a MCS, in which the terminal event in the sequence indicates whether or not all components in the MCS are in the fail state. For a series system, there are N_c FPSs, one corresponding to each component. For a parallel system, there is only one MCS and thus one FPS. A FPS is comprised of a chain of *failure path events* (FPEs), each of which is associated with a component and gives the state of the sequence up to that event. Let $E_{f,i}$ be the node in the BN that represents the FPE associated with component i. The state of $E_{f,i}$ is defined as

$$\begin{aligned} E_{f,i} &= 0 \quad \text{if} \left\{E_{f,\mathrm{Pa}(i)} = 0\right\} \cap \{C_i = 0\} \\ &= 1 \quad \text{otherwise} \end{aligned} \tag{9}$$

where $E_{f,\mathrm{Pa}(i)}$ defines the state of the FPE node that is parent to $E_{f,i}$. For a series system, the BN formulation using FPSs takes the form shown in Figure 9a, which has the undesirable converging structure. For a parallel system, the BN formulation takes the chain form shown in Figure 9b. These findings suggest that a combination of SPS and FPS formulations can be used to efficiently model general systems. This approach is described in the next section.

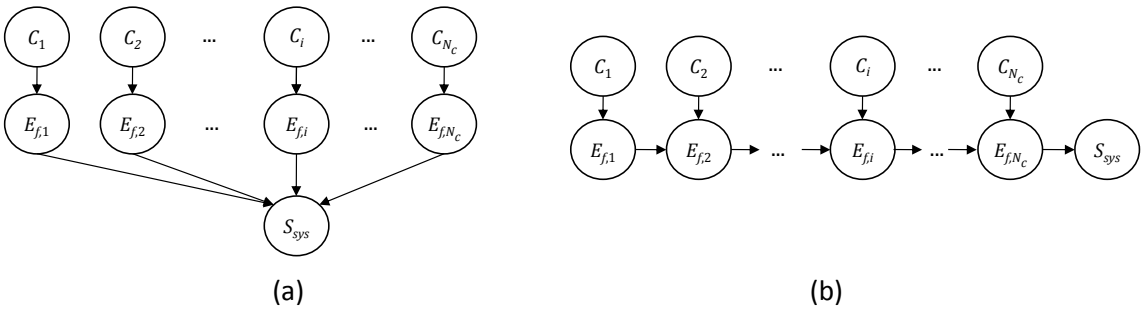


Figure 9: BN using FPEs to model performance of (a) series system, (b) parallel system

5.3 General systems with binary component states

A MLS is a series system of its constituent components. Therefore, using the above formulation, each MLS of the system can be described by an SPS, resulting in a chain-like BN structure. Consider the example system in Figure 10a, which has four MLSs: MLS_1={1,7,8}, MLS_2 ={2,7,8}, MLS_3 ={3,7,8} and MLS_4 ={4,5,6,7,8}. In Figure 10b each MLS is modeled as an individual SPS. The SPEs, $E_{s,i}^{j}$, in each SPS are indexed by a subscript corresponding to the associated component i and a superscript corresponding to the associated MLS j. The dependence between SPEs that share a component is modeled through a common parent node. The system node is in the survival state if the terminal node of any SPS is in the survival state. For reference, the BN formulation in which the MLSs are arranged in chain structures is named *efficient MLS BN formulation*. Similar logic leads to the creation of an *efficient MCS BN formulation*, whereby strings of FPSs are constructed corresponding to each MCS.

The dependence between SPEs or FPEs sharing a component increases the computational demand when performing inference in the BN. By coalescing common SPEs/FPEs that appear in multiple SPSs/FPSs, the number of nodes and links in the BN, and hence the computational demand, are reduced. In the example system, components 7 and 8 appear in all SPSs. We take advantage of this observation and introduce only one "instance" of the SPEs associated with these components. The resulting BN is shown in Figure 11a. Note that this configuration also avoids the converging structure of the system node. It does, however, require a converging SPE node (node $E_{s,7}$ in Figure 11a). The states of such SPE nodes having multiple SPEs as parents are specified using the Boolean relation

$$\begin{aligned} E_{s,i} &= 1 \quad \text{if } \left[\cup \{E_{s,\mathrm{Pa}(i)} = 1\}\right] \cap \{C_i = 1\} \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{10}$$

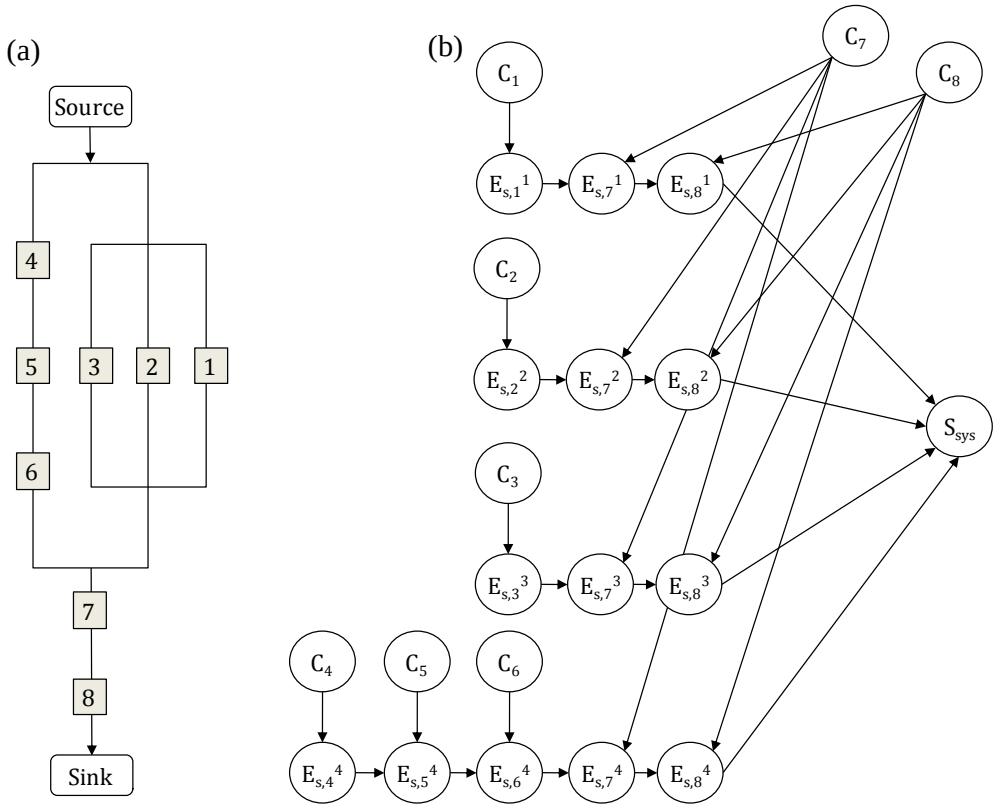


Figure 10: (a) Example system; (b) Efficient MLS formulation with distinct SPSs

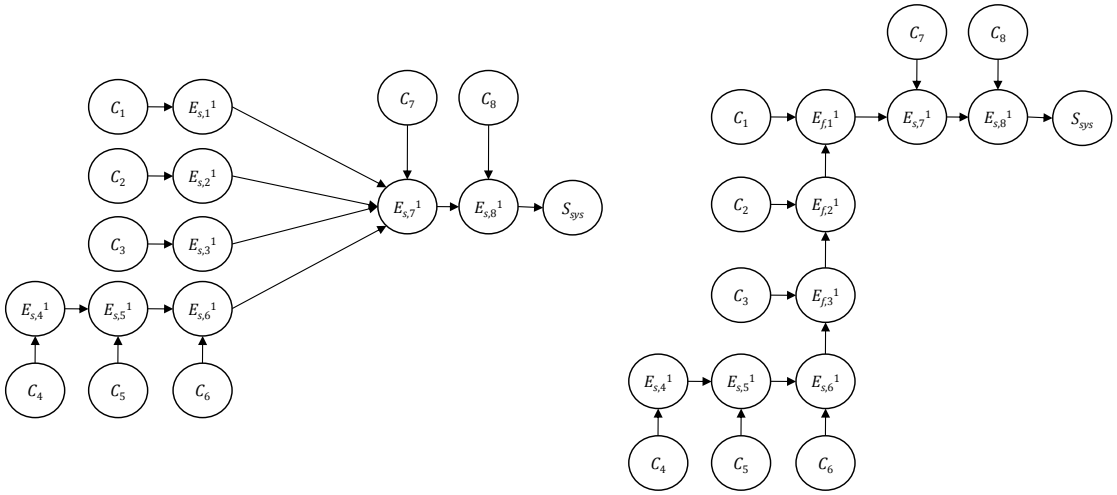


Figure 11: Efficient MLS formulations for the example system with coalesced SPEs associated with components 7 and 8 using (a) a converging structure (b) a chain structure

A notational change has been introduced in Figure 11a: the superscript on each SPE node, which previously indicated the MLS index, now represents the instance of the SPE, i.e. when multiple SPEs are associated with the same component, then they are recognized as different instances of the SPE and are distinguished through the superscript. For this example system, because each component is associated with only one SPE, all superscripts in Figure 11a are 1. In the following, we drop the instance index, unless it is required for clarity.

Note that the converging structure of node $E_{s,7}$ in Figure 11a arises because of the need to represent a parallel sub-system. Since parallel systems are ideally represented using chains of FPEs, the converging structure can be modified by replacing the SPE nodes associated with components 1, 2 and 3 with FPE nodes arranged in a chain, resulting in the BN in Figure 11b. The definition of the FPE nodes with SPE nodes as parents follows the original definition:

$$\begin{aligned} E_{f,i} &= 0 \quad \text{if} \left\{E_{f,\mathrm{Pa}(i)} = 0\right\} \cap \left\{E_{s,\mathrm{Pa}(i)} = 0\right\} \cap \{C_i = 0\} \\ &= 1 \quad \text{otherwise} \end{aligned} \tag{11}$$

where $E_{f,\mathrm{Pa}(i)}$ and $E_{s,\mathrm{Pa}(i)}$ are the FPE and SPE nodes, respectively, that are parents to $E_{f,i}$.

While all three BN models of the example system (Figures 10b, 11a and 1b) are termed efficient MLS formulations, their efficiency is in fact quite variable. The total clique size associated with the BN in Figure 10b is 224, the one associated with the BN in Figure 11a is 108 and the one associated with the BN in Figure 11b is 64. This illustrates the potential efficiency gain when coalescing multiple instances of SPEs in different SPSs.

Thus far, the SPEs in a SPS (FPEs in a FPS) corresponding to a particular MLS (MCS) have been arranged in an arbitrarily selected order. For complex systems, the arrangement of the SPEs in the SPSs may strongly influence our ability to coalesce multiple instances of SPEs in different SPSs (analogously FPEs in FPSs). The order in which SPEs (FPEs) appear can be optimized such that SPEs in as many SPSs (FPEs in as many FPSs) as possible are coalesced. As demonstrated earlier, this reduces the number of nodes and links in the BN. This optimization problem is described next. For brevity, only the formulation employing SPSs is presented; a dual formulation applies to FPSs. Since the focus is on SPEs only, the possibility

to combine SPEs and FPEs is not considered. Such a combination can be performed as an additional step after the optimization of the SPEs (or FPEs), similar to the example shown in Figure 11b.

6. Optimal ordering of survival path events

The aim of this section is to formalize the problem of finding the optimal arrangement of SPEs in SPSs for a general system. Let $L(i^m, j^n) = 1$ indicate the existence of a directed link from node $E_{s,i}^m$ to node $E_{s,j}^n$ in the efficient MLS BN formulation and $L(i^m, j^n) = 0$ indicate the absence of such a link, where i and j are component indices and m and n are indices denoting the instances of these SPE nodes in the BN. Also, let $C_i^m = 1$ indicate a directed link between the node representing component i and node $E_{s,i}^m$ and $S_i^m = 1$ indicate a directed link between $E_{s,i}^m$ and the system node (with $C_i^m = 0$ and $S_i^m = 0$ respectively denoting their absences). The decision variables in the optimization problem are the links between the SPE nodes, $L(i^m, j^n)$. The remaining links, C_i^m and S_i^m, follow from the $L(i^m, j^n)$, as described later. Formulation of this optimization problem assumes the use of only SPE nodes as defined in Eq. (10) and a converging structure at the system node. To further increase computational efficiency of the resulting BN, the converging structure at the system or any other node can be replaced by a chain structure by use of FPEs in the manner described in Figure 11.

Using the number of links in the BN as a proxy for computational demands required when performing inference, the objective of the optimization problem is to minimize the number of links in the BN. This is formulated as

$$\min \left[\sum_{i=1}^{N_c} \sum_{j=1}^{N_c} \sum_{m=1}^{N_I} \sum_{n=1}^{N_I} L(i^m, j^n) + \sum_{i=1}^{N_c} \sum_{m=1}^{N_I} C_i^m + \sum_{i=1}^{N_c} \sum_{m=1}^{N_I} S_i^m \right] \tag{12}$$

where N_I is the maximum number of instances of any SPE. It is desirable that N_I be as small as possible, but its value is not known prior to solving the optimization problem. Thus, an iterative procedure is used to find the smallest N_I for which the optimization problem has a feasible solution. To ensure that the BN structure represents the system through SPEs in the manner described in the preceding section, a number of constraints on the optimization problem are formulated, as follows.

First, every SPE node must have a link pointing to it from the corresponding component node. Specifically, $C_i^m = 1$ if node $E_{s,i}^m$ exists in the BN, which occurs if the decision variables indicate a link going into or out of node $E_{s,i}^m$. (A node without links going into or out of it can be removed from the BN.) Mathematically, this is formulated as a constraint on the optimization problem, written as

$$\left[\sum_{j=1}^{N_c} \sum_{n=1}^{N_I} \{ L(i^m, j^n) + L(j^n, i^m) \} \geq 1 \right] \Rightarrow C_i^m = 1 \tag{13}$$

Second, if $E_{s,i}^m$ exists and has no other SPE node as a child, then it is a terminal node in a SPS. Such a node must have a link to the system node, i.e. $S_i^m = 1$. This leads to the second constraint on the optimization problem, written as

$$\left[\sum_{j=1}^{N_c} \sum_{n=1}^{N_I} L(j^n, i^m) \geq 1 \right] \cap \left[\sum_{j=1}^{N_c} \sum_{n=1}^{N_I} L(i^m, j^n) = 0 \right] \Rightarrow S_i^m = 1 \tag{14}$$

Well known techniques are available for modeling "if-then" propositions, as in the preceding two equations, into constraints in numerical optimization, e.g., see Sarker & Newton (2008).

Third, there are two constraints governing the arrangement of the SPE nodes in the BN: (a) each MLS must be represented by a SPS, and (b) no SPS may exist that is not strictly a MLS. Violation of the first constraint results in exclusion of one or more MLSs, causing underestimation of the system reliability. Violation of the second constraint results in a BN that includes one or more fictitious MLSs, thus causing overestimation of the system reliability.

Constraint (a) requires that each MLS be represented as a SPS, i.e. at least one *permutation* of the SPEs associated with the components in the MLS must be connected as a chain. Let $N_{MLS,i}$ denote the number of components in MLS_i. For the example system in Figure 10a, we have $N_{MCS,1} = N_{MCS,2} = N_{MCS,3} = 3$ and $N_{MCS,4} = 5$. Let P_i denote the set of permutations, without replacement, of the component indices in MLS_i and define $p_i^\alpha = \{p_{i,1}^\alpha, p_{i,2}^\alpha, \dots, p_{i,N_{MCS,i}}^\alpha\}$ as its α^{th} member. As an example, for the system in Figure 10a, we have $P_1 = \{p_1^1 = (1,7,8),\ p_1^2 = (1,8,7),\ p_1^3 = (8,1,7),\ p_1^4 = (7,1,8),\ p_1^5 = (7,8,1),\ p_1^6 = (8,7,1)\}$.

Next, let Q_i denote the set of permutations with replacement of $N_{MCS,i}$ draws from the index set $\{1, \dots, N_I\}$ and define $q_i^\beta = \{q_{i,1}^\beta, q_{i,2}^\beta, \dots, q_{i,N_{MLS,i}}^\beta\}$ as its β^{th} member. Using the example in Figure 10a and assuming $N_I = 2$, we have $Q_1 = \{q_1^1 = (1,1,1),\ q_1^2 = (1,1,2),\ q_1^3 = (1,2,1),\ q_1^4 = (1,2,2),\ q_1^5 = (2,1,1),\ q_1^6 = (2,1,2),\ q_1^7 = (2,2,1),\ q_1^8 = (2,2,2)\}$. Note that P_i has $N_{MLS,i}!$ members, while Q_i has $N_I^{N_{MLS,i}}$ members.

Define the set $r_i^{\alpha,\beta} = [r_{i,1}^{\alpha,\beta}, r_{i,2}^{\alpha,\beta}, \dots, r_{i,N_{MLS,i}}^{\alpha,\beta}]$, which combines the elements of p_i^α and q_i^β. Specifically, $r_i^{\alpha,\beta}$ includes the component indices in p_i^α with superscripts specified according to q_i^β. For the example system, $r_1^{1,1} = \{1^1, 7^1, 8^1\}$, $r_1^{1,2} = \{1^1, 7^1, 8^2\}$, $r_1^{2,4} = \{1^1, 8^2, 7^2\}$, etc. Overall, for this particular MLS, there are $3! * 2^3 = 48$ possible ways to arrange the component indices and the instance superscripts.

For convenience, define the sum

$$X_i^{\alpha,\beta} = \sum_{l=1}^{N_{MLS,i}-1} L[r_{i,l}^{\alpha,\beta}, r_{i,l+1}^{\alpha,\beta}], \tag{15}$$

where $r_{i,l}^{\alpha,\beta}$ is the l^{th} element of $r_i^{\alpha,\beta}$. $X_i^{\alpha,\beta} = N_{MLS,i} - 1$ only if the SPEs corresponding to the components in MLS_i form a SPS in the order specified by p_i^α and with instance indices according to q_i^β. For a SPS associated with the ith MLS to exist in the BN, we must have $X_i^{\alpha,\beta} = N_{MLS,i} - 1$ for *at least one* component/instance index ordering from the set $r_i^{\alpha,\beta}$. The constraint is written as

$$\max_{\alpha,\beta} X_i^{\alpha,\beta} \geq N_{MLS,i} - 1 \qquad \forall i\,, \qquad \alpha = 1, \dots, N_{MLS,i}!, \qquad \beta = 1, .., N_I^{N_{MCS,i}} \tag{16}$$

where, $\geq$ is used rather that than equality for convenience of the optimization algorithm.

Constraint (b) requires that no SPS exist in the BN that does not correspond to a MLS. Consider the BN segment shown in Figure 12a. Let the shaded nodes ($E_{s,1}^1 \rightarrow E_{s,2}^1 \rightarrow E_{s,3}^1 \rightarrow E_{s,4}^1$) represent a particular permutation of component/instance indices $r_i^{(\alpha,\beta)} =$

$\{1^1, 2^1, 3^1, 4^1\}$ resulting in a valid SPS. Constraint (b) must prohibit a SPE $E_{s,j}^n$, for any n, from "branching-off" the SPS at any node, i.e. being a child of any node in the chain, unless the component j exists in a MLS with all the preceding components in the sequence. For example, in Figure 12a, $E_{s,j}^n$ cannot exist as a child of $E_{s,3}^1$ unless components 1, 2, 3 and j exist together in a MLS. If components 1, 2, 3 and j do not exist in a MLS, then the false survival path shown by nodes with dashed edges is introduced into the BN. The associated constraints for all valid SPSs are written as

$$\left\{\left[X_i^{\alpha,\beta} = N_{MLS,i} - 1\right] \Rightarrow L\left[r_{i,l}^{\alpha,\beta}, j^n\right] = 0, \forall j: \left\{p_{i,1}^{\alpha}, \dots, p_{i,l}^{\alpha}, j\right\} \notin MLS_m, \forall m\right\}, \quad \forall i, n, l, \alpha\beta \tag{17}$$

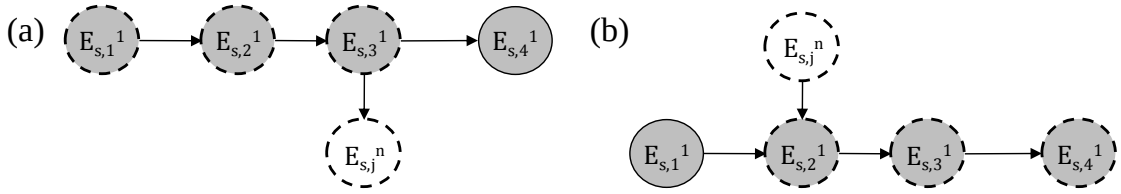


Figure 12: BN used to illustrate constraint (b)

Furthermore, constraint (b) must prohibit SPE $E_{s,j}^n$, for any n, from being a parent to any node in a valid SPS, unless component i exists in a MLS with all subsequent components in the sequence. For example, in Figure 12b, $E_{s,j}^n$ cannot be a parent of $E_{s,2}^1$ unless components 2, 3, 4 and j exist together in a MLS. The second constraint for all valid SPSs takes the form

$$\begin{gathered}\left\{\left[X_i^{(\alpha,\beta)} = N_{MLS,i} - 1\right] \Rightarrow \right. \\ \left. L\left[j^n, r_{i,l}^{(\alpha,\beta)}\right] = 0, \quad \forall j: \left\{j, p_{i,l}^{(\alpha)}, \dots, p_{i,N_{MLS,i}}^{(\alpha)}\right\} \notin MLS_m, \forall\, m\right\}, \quad \forall i, n, l, \alpha, \beta\end{gathered} \tag{18}$$

Combining (17) and (18) results in constraint (b), written as

$$\begin{gathered}\left\{\left[X_i^{(\alpha,\beta)} = N_{MLS,i} - 1\right] \Rightarrow \right. \\ \sum_{m=1, m\neq i}^{N_{MLS}} \sum_{n=1}^{N_I} \sum_{l=1}^{N_{MLS,i}} \sum_{\forall j: \left\{p_{i,1}^{(\alpha)}, \dots, p_{i,l}^{(\alpha)}, j\right\} \notin MLS_m} L\left[r_{i,l}^{(\alpha,\beta)}, j^n\right] + \\ \left. \sum_{m=1, m\neq i}^{N_{MLS}} \sum_{n=1}^{N_I} \sum_{l=1}^{N_{MLS,i}} \sum_{\forall j: \left\{j, p_{i,l}^{(\alpha)}, \dots, p_{i,N_{MLS,i}}^{(\alpha)}\right\} \notin MLS_m} L\left[j^n, r_{i,l}^{(\alpha,\beta)}\right] = 0 \right\}, \quad \forall i, \alpha, \beta\end{gathered} \tag{19}$$

Constraint (b) along with the objective function, the minimization of which ensures that links that are not necessary for constructing the required SPSs are not in the BN, prohibits formation of invalid SPSs in the BN.

The efficient MCS formulation is the dual of the efficient MLS formulation, where strings of FPSs are constructed corresponding to each MCS. Therefore, for constructing an FPS formulation, an optimization problem identical to the above with SPEs replaced by FPEs can be formulated.

The integer optimization problem described above requires consideration of permutations of component indices and instances. Consequently, the size of the problem rapidly grows with

the number of components. To overcome this difficulty, several heuristics are introduced in a later section of this paper.

6.1 Extension to multi-state flow systems

As with the standard MCS formulation, the efficient MCS formulation can be adapted to handle multi-state problems through the application of the Max-Flow Min-Cut Theorem. The topology of the BN need not differ from the topology used for the binary state problem. It is only necessary to increase the number of states associated with each node and use arithmetic expressions instead of Boolean relations to define the CPTs. The states of the FPE nodes are associated with capacity values defined by

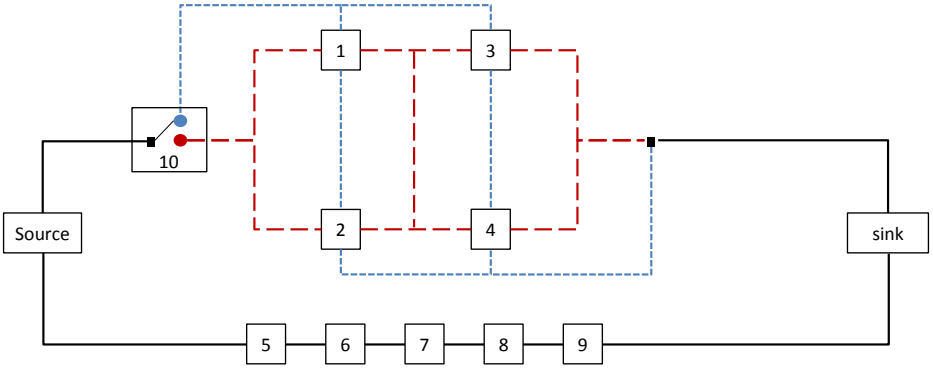
$$Cap_{E_{f,i}} = \min_{E_{f,j} \in \mathrm{Pa}(E_{f,i})} Cap_{E_{f,j}} + Cap_i \tag{20}$$

That is, the capacity value assigned to node $E_{f,i}$ is equal to the minimum of the capacity values of its parent FPE nodes, plus the capacity of the associated component. Thus, the capacity of each FPE node can be thought of as representing the "running total" of the capacities of the MCSs that it is a part of. The capacity of node S_{sys}, representing the maximum operating level of the system, is the minimum capacity among all MCSs. Thus, it is defined as

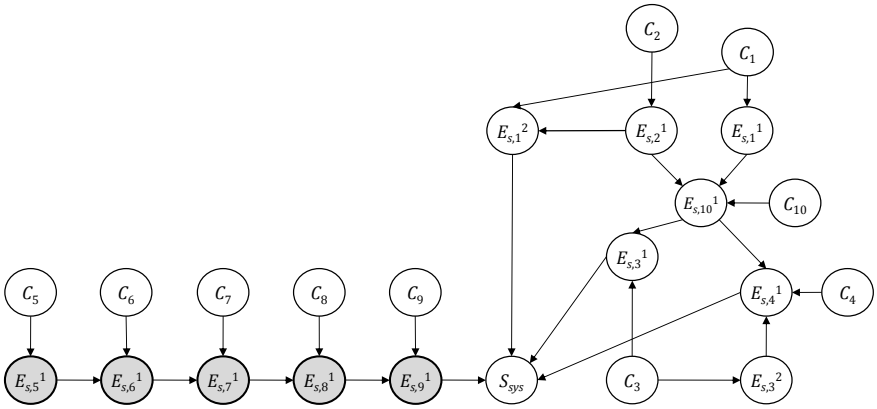
$$Cap_{sys} = \min_{E_{f,j} \in \mathrm{Pa}(Sys)} Cap_{E_{f,j}} \tag{21}$$

6.2 Example Application

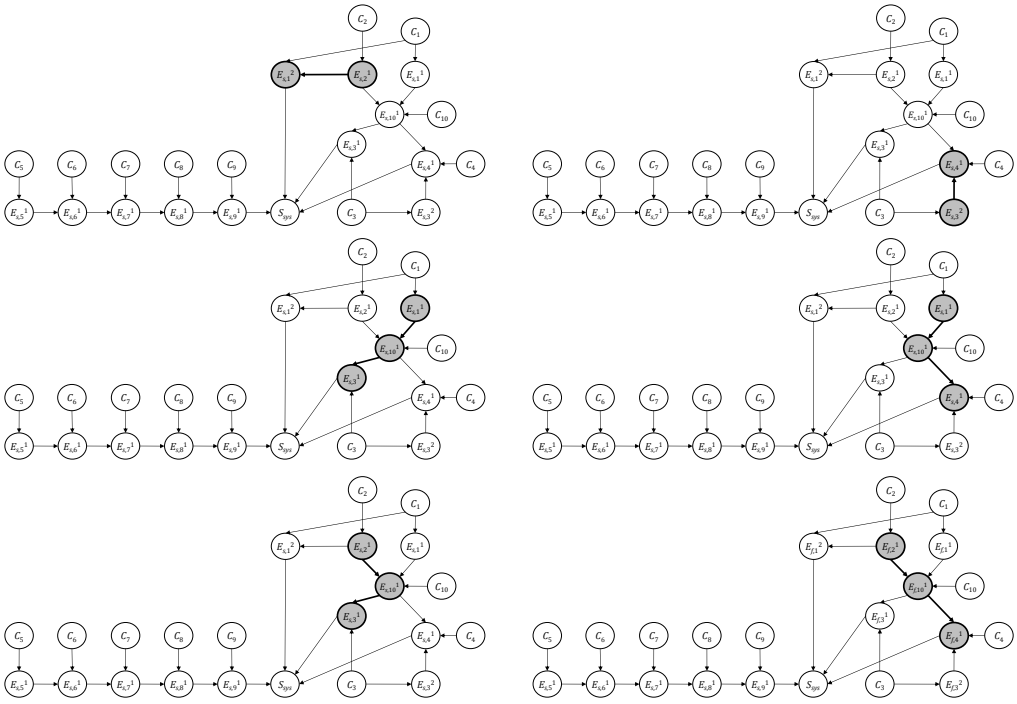
To illustrate the computational advantages of using the proposed efficient BN formulation, consider the system in Figure 13a consisting of 10 labeled components. In this example, a source and a sink are connected, by default, by two connectivity paths. One connectivity path is represented by a solid line and the other by short dashed lines. In this configuration, the system has three MLSs: {1,2}, {3,4}, and {5,6,7,8,9}. Component 10 represents a switch, which can be used to change from the default configuration to an alternate configuration represented by long dashed lines in Figure 13a. This alternate configuration can be "switched to," provided component 10 is working. This adds 4 more connectivity paths through the system:{1,3,10}, {1,4,10}, {2,3,10}, {2,4,10}. Thus, overall, the system has 7 MLSs. The BN obtained using the proposed optimization algorithm is shown in Figure 13b, where the SPS corresponding to the MLS {5,6,7,8,9} is highlighted by shaded nodes. Figure 13c shows the same BN with the SPSs corresponding to each of the remaining MLSs similarly highlighted. The total clique size associated with this BN is 184. The total clique size of the MLS BN formulation with the converging structure (similar in form to the general configuration in Figure 4) is 5,140, and it is $2^{11} = 2048$ with the naïve BN formulation. When intermediate nodes are introduced in the MLS BN formulation to ensure no node has more than three parents (a common approach for reducing the number of parents in BNs, see Straub and Der Kiureghian 2010a), the total clique size reduces to 804, but remains substantially higher than that achieved with the efficient MLS BN topology. Thus, the optimized BN is over an order of magnitude computationally more efficient than the alternate approaches. Furthermore, note that node S_{sys} is a parallel system of its four parent SPS nodes. A small additional computational advantage is achieved by modifying the converging structure with a chain structure, as demonstrated in Figure 11b for the earlier example application.



(a)



(b)



(c)

Figure 13: (a) Example network system; (b) efficient MLS BN formulation with one SPS highlighted; (c) BNs showing SPSs corresponding to remaining MLSs

7 Heuristic Augmentation

As mentioned earlier, the binary optimization problem described above rapidly grows in size due to the requirement to consider all permutations of component indices and instances. To overcome this problem, in this section we present two heuristics: One aims at reducing the number of components that need to be considered, and one aims at reducing the number of permutations. Additional heuristics can be developed to further improve the performance and scalability of the optimization algorithm described in this paper.

7.1 Heuristic employing super components

Der Kiureghian and Song (2008) introduced the idea of multi-scale modeling of systems, whereby subsets of elementary components are grouped into "super components." Analysis is performed for individual super components and results are then aggregated at the system level. The super components typically comprise simple sub-systems, such as components that exist in series or in parallel along a system link. Use of super components reduces the effective number of components and, thereby, permutations of their indices. It also facilitates the combining of SPSs and FPSs.

Once again, consider the simple system shown in Figure 10a. Components C_4, C_5 and C_6 exist in series as do components C_7 and C_8. We replace these subsets of components by two super components SC_1 and SC_2, respectively, as shown in Figure 14. The system still has 4 MLSs, but the number of components in them is reduced: {1,SC_2}, {2,SC_2},{3, SC_2},{SC_1,SC_2}.

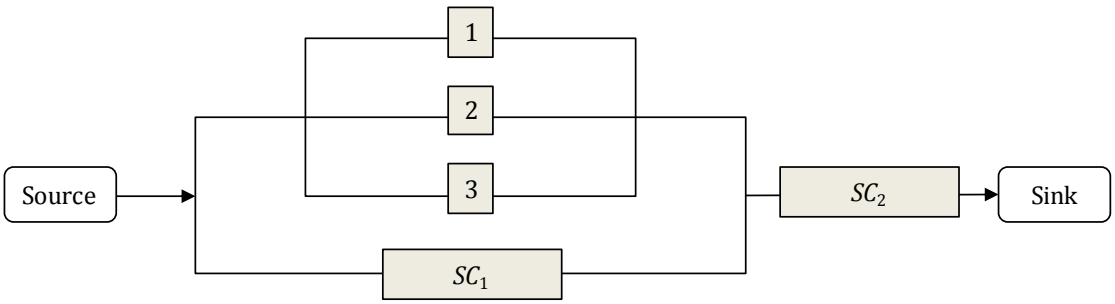


Figure 14: System in Figure 10a with component sets (C_4, C_5, C_6) and (C_7, C_8) respectively replaced by super components SC_1 and SC_2

Examination of Figure 14 reveals that components C_1, C_2, C_3 and SC_1 exist in parallel. These components are next replaced by a single super component resulting in the system representation shown in Figure 15. Now, components SC_2 and SC_3 exist in series and can be replaced by another super component. The BN resulting from this sequential procedure for identifying components that may be grouped and replaced by a super component is shown in Figure 16. For super components containing less than 4 constituent components, a converging structure is used. For super components with 4 or more constituent components, a chain structure is utilized.

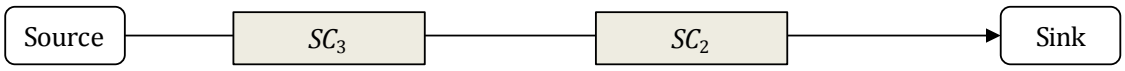


Figure 15: System in Figure 14 with components (C_1, C_2, C_3, SC_1) replaced by a super component

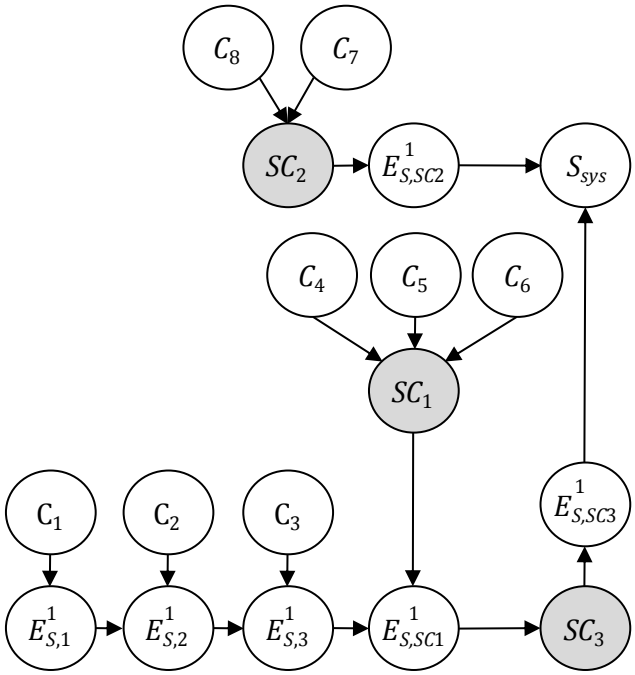


Figure 16: BN constructed for system in Figure 10a using the super component heuristic

Note that components in a super component need not be contiguous. For example, in the system in Figure 17, components 1 and 4 can be put into a super component because, with regard to formation of MLSs and MCSs, they have the same effect as if they physically existed in series.

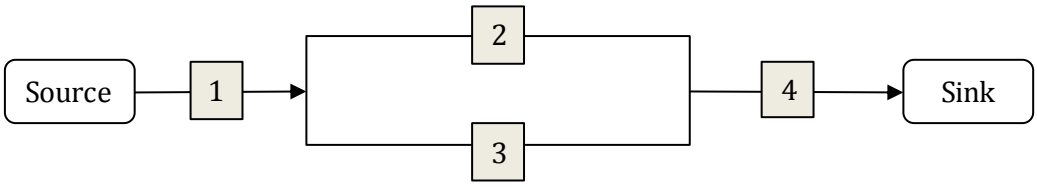


Figure 17: Example system illustrating non-contiguous components that can be combined into a super component.

We now present an algorithm for sequential identification and replacement of elementary components by super components, or sets of super components by higher level super components. The first step in the algorithm is construction of an initial "correspondence" matrix $\mathbf{M}^0$ that contains a row corresponding to each MLS (MCS) and a column corresponding to each elementary component. The elements M^0_{ij} of this matrix are defined by

$$M_{i,j}^{0} = 1 \quad \text{if component } j \text{ is a member of MLS (MCS) } i$$
$$= 0 \quad \text{otherwise} \tag{22}$$

For the example system in Figure 10a, matrix $\mathbf{M}_0$ is as given at the top of Figure 18. In each subsequent step of the algorithm, the correspondence matrix is updated by eliminating columns corresponding to components that are grouped into a super component, and creating a new column to represent the new super component. Let $\mathbf{M}^p$ denote the correspondence matrix in the pth step of the algorithm with its elements denoted $M_{i,j}^{p}$.

Two types of super components are identified. Class A super components are made up of groups of components (or previously formed super components) that always appear together in a MLS (MCS) and never appear separately. In a MLS(MCS)-based formulation, Class A super components correspond to components that exist in series (parallel). To identify such super components in the p^{th} iteration of the algorithm, we assign to each component (or previously formed super component) j the quantity $m_j^{(A,p)}$ defined by

$$m_j^{A,p} = \sum_{i=1}^{N_{MLS}} 2^i \, M_{i,j}^{p} \tag{23}$$

This quantity is identical for different components only when the components always appear together in certain MLSs (MCSs) and never separately in different MLSs (MCSs). For example, for the system in Figure 10a, we have $m_1^{A,0} = 2$, $m_2^{A,0} = 4$, $m_3^{A,0} = 8$, $m_4^{A,0} = m_5^{A,0} = m_6^{A,0} = 16$ and $m_7^{A,0} = m_8^{A,0} = 30$. Each set of components having identical $m_j^{A,p}$ can be grouped to form a Class A super component. Matrix $\mathbf{M}^p$ must then be updated to $\mathbf{M}^{p+1}$ by removing columns corresponding to components that were grouped and adding a column to represent the new super component. Let SC_j denote the newly formed super component. The elements of the new column of the updated matrix are defined by

$$M_{i,SC_j}^{p+1} = 1 \quad \text{if} \sum_{k \in C_{grp}^{p}} M_{i,k}^{p} > 0$$
$$= 0 \quad \text{otherwise} \tag{24}$$

where C_{grp}^{p} is the set of components that were grouped to form the new super component. We perform this operation repeatedly until all super components of Class A are formed. For the example system in Figure 10a, using an MLS formulation, two super components of Class A are identified and the updated matrices $\mathbf{M}^1$ and $\mathbf{M}^2$ are as shown in the middle of Figure 18.

$\mathbf{M^0}$

	Comp 1	Comp 2	Comp 3	Comp 4	Comp 5	Comp 6	Comp 7	Comp 8
MLS 1	1	0	0	0	0	0	1	1
MLS 2	0	1	0	0	0	0	1	1
MLS 3	0	0	1	0	0	0	1	1
MLS 4	0	0	0	1	1	1	1	1

$\mathbf{M^1}$

	Comp 1	Comp 2	Comp 3	Comp 7	Comp 8	SC_1
MLS 1	1	0	0	1	1	0
MLS 2	0	1	0	1	1	0
MLS 3	0	0	1	1	1	0
MLS 4	0	0	0	1	1	1

$\mathbf{M^2}$

	Comp 1	Comp 2	Comp 3	SC_1	SC_2
MLS 1	1	0	0	0	1
MLS 2	0	1	0	0	1
MLS 3	0	0	1	0	1
MLS 4	0	0	0	1	1

$\mathbf{M^3}$

	SC_2	SC_3
MLS 1	1	1

Figure 18: Correspondence matrices of example system before and after identification and formation of Type A and Type B super components.

The second class of super components, Class B, comprises components that appear in different MLSs (MCSs), but share these MLSs with the same set of other components. For a MLS (MCS)-based formulation, these correspond to components in parallel (series). To identify such super components in the p^{th} iteration of the algorithm, we assign to each component (or previously formed super component) j the quantity $m_j^{B,p}$ defined by

$$m_j^{B,p} = \sum_i \left[M_{i,j}^p * \left(\sum_{k \neq j} 2^k M_{i,k}^p \right) \right] \tag{25}$$

One can easily verify that components (or super components) that have identical $m_j^{B,p}$ values satisfy the above conditions for a Class B super component and can be so grouped. As an example, for the system in Figure 10a, we have $m_1^{B,2} = m_2^{B,2} = m_3^{B,2} = m_{SC_1}^{B,2} = 32$. Matrix $\mathbf{M}^p$ is now updated by removing columns corresponding to the grouped components and adding a column for the new super component with elements defined by Eq. (24). Since the combined components come from different MLSs (MCSs), this process renders some rows of the matrix zero, which can be removed. For the example system, this process leads to the updated correspondence matrix $\mathbf{M}^3$, which is shown in the bottom of Figure 18.

The above iterative procedure for finding and replacing components with super components of Class A and Class B is repeated until no super components remain. The matrix $\mathbf{M}^p$ that corresponds to the last iteration is then used to specify the components and the MLSs or MCSs required for defining the optimization problem.

7.2 Heuristic for reducing the number of permutations

Recall that the optimization algorithm considers all permutations of component indices and instances as specified in the sets $r_i^{\alpha,\beta} = \left[r_{i,1}^{\alpha,\beta}, r_{i,2}^{\alpha,\beta}, \ldots, r_{i,N_{MLS,i}}^{\alpha,\beta}\right],\ i = 1, \ldots, N_{MLS}$. (Everywhere in this section the acronym MLS can be exchanged with MCS.) The aim of the second heuristic is to drop selected members of these sets. Observe that if a set of components appear in all MLSs, then it is not necessary to consider permutations of their indices. (In fact, such a set of components would be identified and combined as a super component.). The present heuristic identifies components that appear in several (but not all) MLSs and locks the order of appearance of their indices, thus removing selected members of $r_i^{\alpha,\beta}$. This heuristic is likely to result in a BN topology that is suboptimal, but is necessary if the size of the optimization problem is larger than can be handled by the available resources. The heuristic should be used after reducing the size of the problem through identification of super components.

To facilitate the explanation of the heuristic, an example system is used to illustrate each step of the procedure. Consider an arbitrary system with seven MLSs:

$$MLS_1 = \{1,2,3,4\}$$
$$MLS_2 = \{1,4,5,6,8\}$$
$$MLS_3 = \{1,4,7\}$$
$$MLS_4 = \{2,3,5\}$$
$$MLS_5 = \{1,5,7\}$$
$$MLS_6 = \{1,2,9\}$$
$$MLS_7 = \{7,9\}$$

Step 1: Create an ordered list of the component indices based on the number of times they appear in the various MLSs. Call this list O. For the example system, the number of occurrences of each component within a MLS is shown in the following table:

Component	number of appearances in a MLS
1	5
2	3
3	2
4	3
5	3
6	1
7	3
8	1
9	2

The above table leads to the ordered list $O = \{1,2,4,5,7,3,9,6,8\}$. In the case of ties, the order is based on component index value.

Step 2: Re-order the components within the MLSs based on the order O. For the example system, the new MLS component orders are:

$$MLS_1 = \{1,2,4,3\}$$
$$MLS_2 = \{1,4,5,6,8\}$$
$$MLS_3 = \{1,4,7\}$$
$$MLS_4 = \{2,5,3\}$$

$$MLS_5 = \{1,5,7\}$$
$$MLS_6 = \{1,2,9\}$$
$$MLS_7 = \{7,9\}$$

Step 3a: Determine all pair-wise intersecting sets between the MLSs:

$$M_{i,j} = MLS_i \cap MLS_j, \qquad i = 1, \dots, (N_{MLS} - 1), \qquad j = (i+1), \dots, N_{MLS} \tag{26}$$

For the example system, we have $M_{1,2} = M_{1,3} = M_{2,3} = \{1,4\}$, $M_{1,6} = \{1,2\}$, $M_{3,5} = \{1,7\}$, $M_{2,5} = \{1,5\}$, $M_{1,4} = \{2,3\}$, $M_{1,5}$, $M_{2,4}$, $M_{2,6}$, $M_{3,6}$, $M_{3,7}$, $M_{4,5}$, $M_{4,6}$, $M_{5,6}$, $M_{5,7}$ and $M_{6,7}$ are sets of cardinality 1 and $M_{17} = M_{27} = M_{34} = M_{47} = \emptyset$.

Step 3b: Define G as a set containing the unique sets $M_{i,j}$ with cardinality greater than 1. For the example system, $G = \{\{1,4\},\{1,2\},\{1,5\},\{1,7\},\{2,3\}\}$. This set contains sets of components that appear together in two or more MLSs with the component indices ordered according to Step 1.

Step 4: Sequentially assign to each MLS a set from within G whose intersection with the MLS has the largest cardinality. Let $g_{i,k}$ be the intersection of the i^{th} MLS with the k^{th} member of G, G_k, i.e.,

$$g_{i,k} = MLS_i \cap G_k \tag{27}$$

Define g_i' as the set $g_{i,k}$ which has the largest cardinality:

$$g_i{}' = \{g_{i,max} : |g_{i,max}| = \max_k |g_{i,k}|\} \tag{28}$$

In the case of ties in the cardinality, the set within G that appears first is selected. $g_i{}'$ is the set assigned to the i^{th} MLS. Once a set from within G has been assigned to a MLS, place it in a new set G', if it has not already been placed there. Define $n'_{c,j}$ as the number of times component j appears within the sets in G'. When a component has appeared in N_I sets within G' (where N_I is the maximum allowed number of instances of SPEs in the optimization problem), remove from G all remaining sets containing that component, unless the set is already a member of G'. This step is necessary because, if $n'_{c,j} > N_I$ for any component, the optimization problem may become infeasible.

For the example system, using $N_I = 2$, the sets within G with the longest intersection with MLS_1 are {1,4}, {1,2}, and {2,3}. Because set {1,4} appears first in G, it is assigned to MLS_1. Therefore, $g_1' = \{1,4\}$ and this set is added to the set of sets G'. It follows that $n'_{c,1} = n'_{c,4} = 1$ and $n'_{c,j} = 0$ for components $j = 2,3,5, \dots, 9$. These results for MLS_1 are summarized as follows:

For $i = 1$, $MLS_1 = \{1,2,4,3\}$:

$$g_1' = \{1,4\};\ G' = \{\{1,4\}\}$$
$$n'_{c,1} = 1, n'_{c,4} = 1, n'_{c,j} = 0, j = 2,3,5, \dots, 9$$
$$G = \{\{1,4\},\{1,2\},\{1,5\},\{1,7\},\{2,3\}\}$$

The results for the remaining MLSs are:

For $i = 2$, $MLS_2 = \{1,4,5,6,8\}$:

$$g_2' = \{1,4\};\ G' = \{\{1,4\}\}$$

$$n'_{c,1} = 1,\ n'_{c,4} = 1, n'_{cj} = 0, j = 2,3,5,\dots,9$$
$$G = \{\{1,4\},\{1,2\},\{1,5\},\{1,7\},\{2,3\}\}$$

For $i = 3,\ MLS_3 = \{1,4,7\}$:

$$g'_3 = \{1,4\};\ G' = \{\{1,4\}\}$$
$$n'_{c,1} = 1, n'_{c,4} = 1, n'_{c,j} = 0, j = 2,3,5,\dots,9$$
$$G = \{\{1,4\},\{1,2\},\{1,5\},\{1,7\},\{2,3\}\}$$

For $i = 4,\ MLS_4 = \{2,5,3\}$:

$$g'_4 = \{2,3\};\ G' = \{\{1,4\},\{2,3\}\}$$
$$n'_{c,1} = 1, n'_{c,4} = 1, n'_{c,2} = 1; n'_{c,3} = 1, n'_{c,j} = 0, j = 5,\dots,9$$
$$G = \{\{1,4\},\{1,2\},\{1,5\},\{1,7\},\{2,3\}\}$$

For $i = 5,\ MLS_5 = \{1,5,7\}$:

$$g'_5 = \{1,5\};\ G' = \{\{1,4\},\{2,3\},\{1,5\}\}$$
$$n'_{c,1} = 2, n'_{c,4} = 1, n'_{c,2} = 1, n'_{c,3} = 1, n'_{c,5} = 1, n'_{c,j} = 0, j = 6,\dots,9$$
$$G = \{\{1,4\},\{1,5\},\{2,3\}\}$$

For $i = 6,\ MLS_6 = \{1,2,9\}$:

$$g'_6 = \emptyset;\ G' = \{\{1,4\},\{2,3\},\{1,5\}\}$$
$$n'_{c,1} = 2, n'_{c,4} = 1, n'_{c,2} = 1, n'_{c,3} = 1, n'_{c,5} = 1, n'_{c,j} = 0, j = 6,\dots,9$$
$$G = \{\{1,4\},\{1,5\},\{2,3\}\}$$

For $i = 7, MLS_7 = \{7,9\}$:

$$g'_7 = \emptyset;\ G' = \{\{1,4\},\{2,3\},\{1,5\}\}$$
$$n'_{c,1} = 2, n'_{c,4} = 1, n'_{c,2} = 1, n'_{c,3} = 1, n'_{c,5} = 1, n'_{c,j} = 0, j = 6,\dots,9$$
$$G = \{\{1,4\},\{1,5\},\{2,3\}\}$$

Note that, when MLS_5 is considered, component 1 appears twice in the set G'. Because component 1 has appeared $N_I = 2$ times, all sets containing component 1 are removed from the set G, unless they already appeared in G'.

Step 5: The last step of the heuristic is to modify the optimization problem so that permutations on components not contained within the sets $g'_i,\ i = 1,\dots,N_{MLS}$, are not considered. Recall that $r_i^{\alpha,\beta} = \left[r_{i,1}^{\alpha,\beta}, r_{i,2}^{\alpha,\beta},\dots, r_{i,N_{MLS,i}}^{\alpha,\beta}\right]$ combines component indices according to set p_i^{α} with instance indices according to the set q_i^{β}. To reduce the size of the optimization problem, we remove from the set p_i^{α} all permutations of component indices that are not consistent with the order specified in g'_i. Furthermore, we remove from the set q_i^{β} all instance indices that are greater than $n'_{c,i}$.

7.3 Example Application

Once again consider the system in Figure 13a consisting of 10 components and 7 MLSs. The super component heuristic is employed to obtain the solution in Figure 13b. The only super component for this system is the one including components 5-9, which are arranged in a single SPS. The arrangement of the remaining SPSs is found through the full optimization algorithm described in section 6. The formulation of Figure 13b was obtained without using

the heuristic described in section 7.2. This idealized network system is sufficiently simple that it is possible to obtain a solution without requiring the heuristic, but for larger systems this may not be case. The BN obtained for this system using the optimization algorithm augmented by the heuristic of section 7.2 is shown in Figure 19. The solution obtained is suboptimal. It contains 29 links, whereas the optimal solution contains 26 links. The total clique size for this BN is 252 (as opposed to 184 for the optimal BN). However, using the Tomlab optimization environment (Holmström 2008), it took over 12 times longer to obtain a solution without the heuristic than when the heuristic was invoked. Thus, in using the heuristic, there is a trade-off between the amount of time required to solve the optimization problem and the optimality of the resulting BN topology. However, even with the heuristic, the efficient MCS formulation is substantially more efficient than the standard MCS formulation.

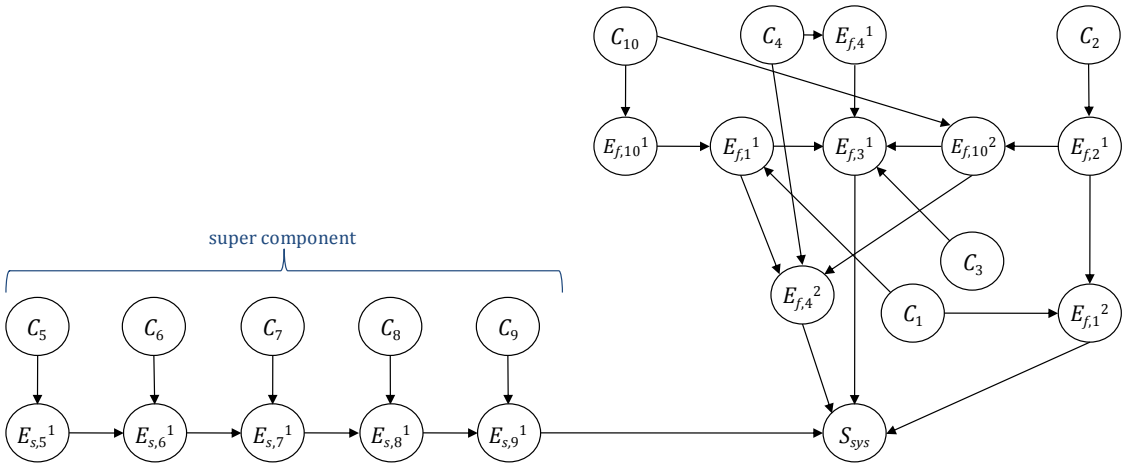


Figure 19: Efficient MCS BN formulation for example system in Figure 13a obtained using the optimization algorithm with both heuristics

8. Conclusions

This paper develops efficient BN formulations for modeling the performance of general systems for which the MLSs or MCSs are known. We show that BNs with nodes in chain structures are more efficient than when nodes are arranged in converging topologies. We demonstrate how BNs with nodes arranged in chain structures are constructed for series and parallel systems. This idea is then extended to develop similar topologies for general systems. Additionally, the idea is extended to consider multi-state flow problems. To automate the construction of efficient BN formulations for general systems, a binary integer optimization program is developed with the goal of automatically constructing the BN such that the number of links is minimized. An example application highlights the advantage gained from the efficient BN formulation. To improve computational efficiency, two heuristic approaches are offered. One employs the concept of super components; the other reduces the number of component permutations that need to be considered. Use of the second heuristic substantially reduces the size of the optimization problem, but may result in a suboptimal BN topology.

9. Acknowledgment

This research was supported by the State of California through the Transportation Systems Research Program of the Pacific Earthquake Engineering Research Center (PEER). Additional support was provided by the Taisei Chair in Civil Engineering at the University of California, Berkeley. The first author also gratefully acknowledges support from a National Science Foundation graduate research fellowship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect those of the funding agencies.

References

Bensi, M., Der Kiureghian, A. & Straub, D. 2011. A Bayesian network methodology for infrastructure seismic risk assessment and decision support. *Report No. 2011/02*, Pacific Earthquake Engineering Research Center, University of California, Berkeley, March.

Bobbio, A. et al., 2001. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. *Reliability Engineering & System Safety*, 71(3), 249-260.

Dechter, R., 1996. Bucket Elimination: A unifying framework for probabilistic inference. *Uncertainty in Artificial Intelligence*, UA1996, 211-219.

Der Kiureghian, A. & Song, J., 2008. Multi-scale reliability analysis and updating of complex systems by use of linear programming. *Reliability Engineering & System Safety*, 93(2), 288-297.

DSL, 2007. *GeNIe 2.0 by Decision Systems Laboratory*, Available at: http://genie.sis.pitt.edu/.

Elias, P., Feinstein, A. & Shannon, C., 1956. A note on the maximum flow through a network. *Information Theory, IRE Transactions on*, 2(4), 117-119.

Ford, L.R. & Fulkerson, D.R., 1956. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3), 399-404.

Friis-Hansen, P., 2004. Structuring of complex systems using Bayesian Networks. In O. Ditlevsen & P. Friis-Hansen, eds. *Proceedings of Two Part Workshop at DTU*. Technical University of Denmark.

Holmström, K., 2008. *Tomlab Optimization Environment*, Available at: http://tomopt.com/tomlab/.

Hugin Expert A/S, 2008. *Hugin Researcher API 7.0*, Denmark: Hugin Expert A/S. Available at: www.hugin.com.

Jensen, F.V. & Nielson, T.D., 2007. *Bayesian Networks and Decision Graphs* second ed., New York: Springer Science+Business Media, LLC.

Kjaerulff, U.B. & Madsen, A.L., 2008. *Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis*, New York: Springer Science+Business Media, LLC.

Lampis, M. & Andrews, J.D., 2009. Bayesian belief networks for system fault diagnostics. *Quality and Reliability Engineering International*, 25(4), 409-426.

Langseth, H. et al., 2009. Inference in hybrid Bayesian networks. *Reliability Engineering & System Safety*, 94(10), 1499-1509.

Lauritzen, S. L., & Spiegelhalter D. J., 1988., Local Computations with Probabilities on Graphical Structures and Their Application to Expert Systems, *Journal of the Royal Statistical Society. Series B (Methodological),* 50(2), 157–224.

Liu, X., Li, H. & Li, L., 2008. Building method of diagnostic model of Bayesian networks based on fault tree. In *Seventh International Symposium on Instrumentation and Control Technology: Sensors and Instruments, Computer Simulation, and Artificial Intelligence*. Beijing, China: SPIE.

Madsen, A.L., 2008. Belief update in CLG Bayesian networks with lazy propagation. *International Journal of Approximate Reasoning*, 49(2), 503-521.

Mahadevan, S., Zhang, R. & Smith, N., 2001. Bayesian networks for system reliability reassessment. *Structural Safety,* 23(3), 231-251.

Nielsen, T.D., Wuillemin, P., Jensen, F.V, Kjaerulff, U., 2000. Using ROBDDs for inference in Bayesian Networks with troubleshooting as an example*, Uncertainty in Artificial Intelligence Proceedings*, p. 426-435.

Pagès, A. & Gondran, M., 1986. *System Reliability*, Springer-Verlag.

Pearl, J., 1988. *Probabilistic reasoning in intelligent systems : networks of plausible inference.* The Morgan Kaufmann series in representation and reasoning, Morgan Kaufmann Publishers, San Mateo, Calif.

Sarker, R.A. & Newton, C.S., 2008. *Optimization modelling: a practical approach*, Boca Raton, Florida: CRC Press, Taylor and Francis Group, LLC.

Shachter, R. D., 1986. Evaluating Influence Diagrams, *Operations Research,* 34(6), 871–882.

Straub, D. & Der Kiureghian, A., 2010a. Bayesian network enhanced with structural reliability methods: Methodology. *Journal of Engineering Mechanics*, ASCE, 136(10):1248-1258.

Straub, D. & Der Kiureghian, A., 2010b. Bayesian network enhanced with structural reliability methods: Application. *Journal of Engineering Mechanics*, ASCE, 136(10):1259-1270.

Torres-Toledano, J. & Sucar, L., 1998. Bayesian Networks for Reliability Analysis of Complex Systems. In *Progress in Artificial Intelligence - IBERAMIA 98.Lecture notes in computer science, vol. 1484*. Springer-Verlag Berlin Heidelberg, pp. 195-206.

Yuan, C. & Druzdzel, M., 2003. An importance sampling algorithm based on evidence pre-propagation. *In Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*.

Yuan, C. & Druzdzel, M., 2006. Importance sampling algorithms for Bayesian networks: Principles and performance. *Mathematical and Computer Modelling*, 43(9-10), 1189-1207.