IASMiRT

SMiRT 26
50 Anniversary Berlin/Potsdam

Transactions, SMiRT-26
Berlin/Potsdam, Germany, July 10-15, 2022
Division VII

COMPUTATIONALLY EFFICIENT ALGORITHMS FOR RISK INFORMED DECISION MAKING IN NUCLEAR PRA

Pragya Vaishanav[1], Saran Srikanth Bodda[2], and Abhinav Gupta[3]

[1]President, Neuratwin Inc., NC, USA (pragya@neuratwin.com)
[2]Research Faculty, CCEE, NCSU, USA
[3]Director, Center for Nuclear Energy Facilities and Structures, NCSU, USA

ABSTRACT

Probabilistic risk assessment (PRA) is used as an essential tool for risk-informed decision making in nuclear industry. The fault and event trees play a crucial role in PRA to estimate the probability of system failure based on the failure probabilities of components. The fault trees or event tree for an actual power plant unit can be fairly large in size with several different types of logic gates, interconnected events, and dependent events, etc. A large fault tree can include hundreds of gates, basic events (BEs), multiple occurring events (MOEs), and dependent events. Complex connectivities can give rise to excessive computational demand and storage requirement for the analysis. Fault and event trees can be solved using the minimal cut-set approaches, or advanced quantification techniques such as Binary decision diagrams or Bayesian networks. However, these techniques can be computationally inefficient for larger fault trees and can run out of memory/storage space. This study focuses on developing and proposing a new approach for accurate estimation of the system-level risk while improving the computational efficiency significantly. More specifically, an attempt is made to reduce the complexity for the analysis of MOEs and dependent events in fault trees. The proposed algorithms in this study present a significant improvement over traditional approaches which makes it highly promising for additional developments.

INTRODUCTION

The nuclear industry has increased its reliance on probabilistic risk assessment (PRA) tools for design, operation, life extension, and regulation. PRA evaluates the risk associated with a specific hazard by a convolution of system fragility and hazard curve (Andersen et al., 2013; Bodda et al., 2019, 2020c; IAEA-Corporate-Author, 2010). A fragility curve of structures, systems, or components (SSCs) is expressed as the conditional probability of failure for a given hazard intensity and is a function of the uncertainties in empirical, experimental, and/or numerical data in the physical and mathematical models of the SSCs (Bodda et al., 2020a,b; Chandrakanth et al., 2019). US Nuclear Regulatory Commission (USNRC) and International Atomic Energy Agency (IAEA) have issued guidelines for conducting a full scope PRA, where the plant level risk is calculated by combining the component and the subsystem fragility curves through a systems analysis (IAEA-Corporate-Author, 2010; US Nuclear Regulatory Commission, 2002). Typically, fault and event trees are used for conducting the systems analysis by logically combining the component level fragilities, and for convoluting the fragilities with the hazard curve (Andersen et al., 2013; Kennedy and Ravindra, 1984).

Fault tree analysis is one of the most powerful tools in PRA to represent the failure probability of a nuclear plant based on the failure probabilities of its basic components. The events in fault and event trees are modeled using binary states: failure or not a failure (success) (RA-S, ASME, 2002). An assessment of

logic trees with n events requires an analysis of 2^n scenarios emerging from the various combinations of event failures. Hence, a complete analysis of a logic tree demands computational complexity of the order 2^n in either time or space. The exponential order of complexity is required mainly to address the dependencies and dependent failure events.

Bottom-up methods have been used when fault trees do not include multiple occurring events (MOEs) and dependent events. However, a complex fault tree typically implements solvers such as minimal cut set analysis (Jung, 2015). The computation of minimal cut sets for fault and event tree is conducted through a widely used MOCUS algorithm (Smith et al., 2016). The computation of the minimal cut sets using this algorithm poses an intractable NP-hard problem (Yeh, 2006, 2021). Therefore, the analysis of large fault trees requires significant computational resources, which makes the analysis of PRA models inefficient and time consuming. In addition, this approach may suffer from exponential complexity for computational time as a function of minimal cut sets in a network when exact computations need to be made. If the network consists of only the OR gates then the time complexity for exact computation would be $O(2^{n_{BE}})$, where n_{BE} is the number of basic events.

There are several traditional methods that reduce the complexity but at the expense of accuracy (Vaishanav et al., 2020). In order to reduce the computational demands for the analysis of the fault and event tree networks, several assumptions or approximations are often relied upon. Binary decision diagram (BDD) is another established method where a fault tree is converted to BDD structure based on the Boolean logic expression of the fault tree (Jung, 2015; Reed, 2017). However, the size of BDD structure increases exponentially with the number of variables. Additionally, the size of BDD structure is very sensitive to the order of variables (Kumar et al., 2010). Therefore, excessive memory demand is an additional limitation for solving large fault trees with BDD algorithm. Consequently, approximations are used and even with approximations, it results in polynomial order of computational complexity (Wegener, 2004). Bayesian networks have also been used for the analysis of the network diagrams (Bodda et al., 2021). However, it appears that all formulations present some computational difficulties (Cavalieri et al., 2017), which are due to either of the following reasons: (i) size of conditional probability tables (Bensi et al., 2013), (ii) the size of clique potentials in the junction-tree algorithm (Kahle et al., 2008), (iii) the recursive algorithm for the identification of minimum link sets (Bensi et al., 2013).

This study focuses on developing and proposing a new approach to solving such problems while improving the computational efficiency significantly. More specifically, an attempt is made to reduce the complexity for the analysis of dependent events in fault trees. The proposed solution is just a beginning and several more scenarios will need to be considered and implemented before this approach can be used for solving true real-world applications. At the same time, it presents a significant improvement over existing implementations which makes it highly promising for the additional developments.

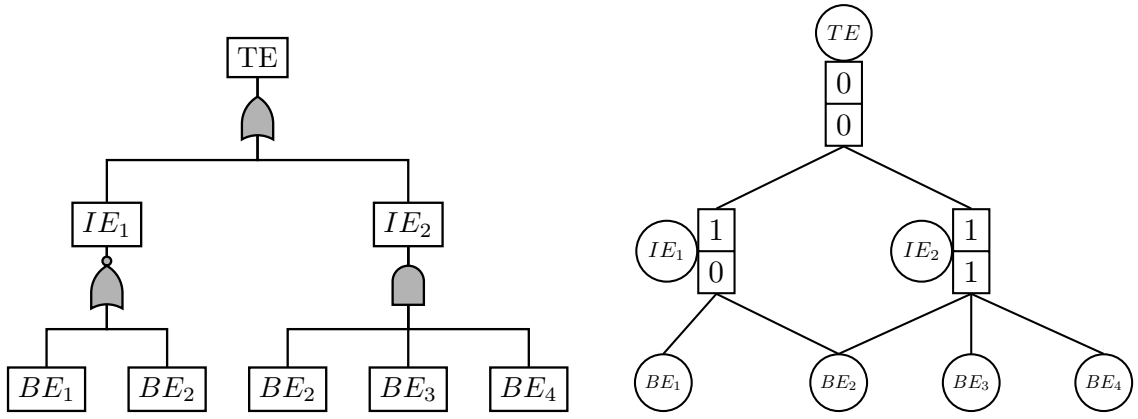
PROPOSED ALGORITHM

In this study, fault trees are represented using a general Tree data structure. Figure 1 (a) shows a generic fault tree and Figure 1 (b) shows the corresponding generic Tree data structure. The following terminologies of Tree data structure are defined to assist with the explanation of the proposed algorithms in the study.

Definitions and Terminology

- ***Top/Root node:*** the top node in a tree data structure.
- ***Child node:*** a node which is a descendant of any node. A node with no child nodes is called as a Leaf node.
- ***Parent node:*** a node which is a predecessor of any node.

- ***Internal node:*** a node which has at least one child node.
- ***Edge:*** a link which connects any two nodes.
- ***Chain/Path:*** a sequence of nodes and edges between two nodes.
- ***Subtree:*** a part of tree data structure which represents a node and all of its descendants.
- ***Height of a node:*** the total number of edges from a particular node to the leaf node in the longest chain. The height of all leaf nodes is zero.
- ***Dependent/common node:*** a node which has more than one parent.
- ***Independent tree:*** a tree with no dependent nodes.
- ***Loop:*** a loop is formed when the start node and the end node of two chains are same.
- ***Logic Gate:*** represents the parent/child relationship. The notation of logic gates are described in detailed in the following section.
- Nodes are represented by circles and the logic gates are denoted by rectangles. The basic event, intermediate event, and top event in the fault tree are referred to as leaf node, internal node, and top node in the Tree data structure.



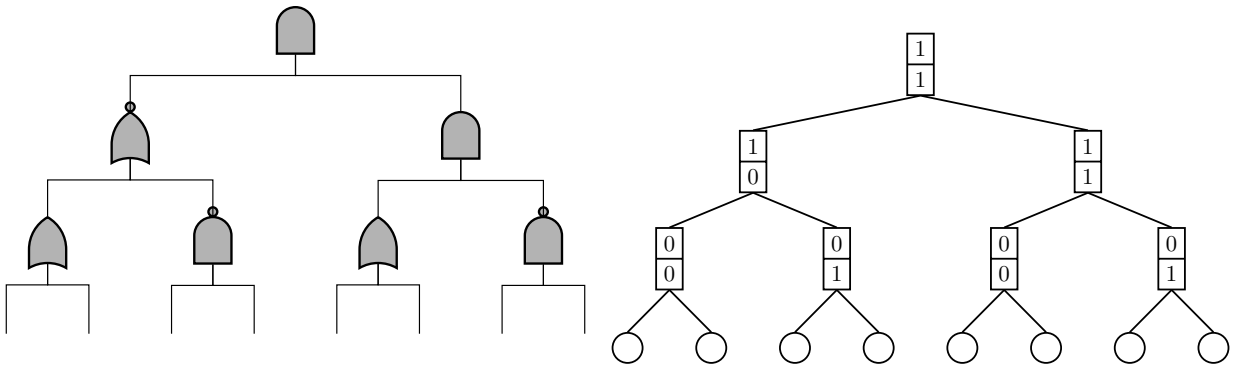
(a) Fault Tree (b) Tree Data Structure

Figure 1. Generic Fault tree and Tree data structure

Compressed Truth Tables

In a fault tree, all the intermediate events and the top event are connected to the basic events through the use of logic gates as shown in Figure 1 (a). In a Tree data structure, the logic gates represent the parent/child relationship, and all the internal nodes have the logic gate attribute. In this study, we examine four logic gates in detail: AND, OR, NAND, and NOR. Table 1 gives the truth table for input nodes A and B when they are connected to an internal node with any of the four logic gates. In the truth tables, the failure and success states of a node are represented using binary states 1 and 0, respectively. The two binary states for nodes A and B lead to 4 (2^2) possible combinations. If suppose n nodes are connected to a logic gate, then there will be 2^n possible combinations of input node states. As seen in Table 1, the highlighted output state is different from the output states of the rest of the three combinations in all the truth tables. This observation

is valid even when n nodes are connected to a logic gate, i.e., output state of only one of the combinations will be different from rest of the $2^n - 1$ combinations.



(a) Fault Tree (b) Tree Data Structure

Figure 2. Logic Gates

Table 1: Truth Tables for logic gates with two input nodes A and B (*The term 0 and 1 in truth table translate to success and failure in fault and event trees*)

AND		
A	B	*Out*
0	0	0
1	0	0
0	1	0
1	1	1

NOR		
A	B	*Out*
0	0	1
1	0	0
0	1	0
1	1	0

OR		
A	B	*Out*
0	0	0
1	0	1
0	1	1
1	1	1

NAND		
A	B	*Out*
0	0	1
1	0	1
0	1	1
1	1	0

Compressed	
Output	1
Input	1

Output	1
Input	0

Output	0
Input	0

Output	0
Input	1

Furthermore, the binary states of each logic gate are mutually exclusive and collective exhaustive. Therefore, if the probability for the shaded output state ($P_{Outputstate}$) of a logic gate is calculated, then the probability for its complement state can be calculated as $1 - P_{Outputstate}$. In this study, we compress the truth table for each logic gate in two states: *input state* and *output state*. Table 1 also shows the *input* state and *output* state for each of the logic gate in the Tree data structure. The logic gates in the fault tree are converted into the *input* state and *output* state in the Tree data structure and an illustration of the same can be seen in Figure 1 (b).

ALGORITHM FOR TREE WITH INDEPENDENT NODES

The binary states probabilities of any internal node in a Tree data structure can be calculated based on its logic gate and the probability of its child nodes. The *output* state and *input* state from the compressed truth table (Table 1) are used to obtain these probabilities. The probabilities *output* state, $P_{Out}(IN)$ and its

complement state, $P_{Out'}(IN)$ for an internal node are calculated using Equation 1.

$$\begin{aligned} P_{Out}(IN) &= \bigcap_{i=1}^{n_c} P_{Inp}(Child_i) \\ P_{Out'}(IN) &= 1 - P_{Out}(IN) \end{aligned} \tag{1}$$

where, Out and Inp are the output and input states of the logic gate corresponding to the internal node IN. $Out^{'}$ is the complement of the output state; i.e., if $Out = 1$ then $Out^{'} = 0$. n_c is the total number of child nodes of the internal node IN. $P_{Inp}(Child_i)$ is the probability of child i. Since all the nodes in the Tree data structure are independent, the intersection of their failure or success state probabilities can be simply obtained by multiplying their respective probabilities as shown in Equation 2.

$$P_{Out}(IN) = \bigcap_{i=1}^{n_c} P_{Inp}(Child_i) = \prod_{i=1}^{n_c} P_{Inp}(Child_i) \tag{2}$$

In this study, the failure probability of the top node is obtained using a bottom-up approach. The approach is illustrated for a simple Tree data structure with independent nodes shown in Figure 3. First, the height of all the internal nodes are calculated. The height of all the leaf nodes is zero. The height of an internal node is calculated by counting the number of edges from the internal node to a leaf node along its longest chain. For example, there are three chains from internal node IN_2 to leaf nodes: $IN_2 - IN_1 - C, IN_2 - IN_1 - D, IN_2 - B$, where, the longest chain is $IN_2 - IN_1 - C$. Hence, the height of IN_2 is equal to 2. Similarly, heights of all the other nodes are calculated and all the internal nodes are sorted in ascending order based on the height as shown in Figure 3. Next, the output probabilities of the internal nodes are calculated one by one in the sorted order $[IN_1, IN_2, TN]$ as shown in Equation 3 using Equation 2..

$$\begin{aligned} P_0(IN_1) &= P_0(C) \times P_0(D), & P_1(IN_1) &= 1 - P_0(IN_1) \\ P_1(IN_2) &= P_0(B) \times P_0(IN_1), & P_0(IN_2) &= 1 - P_1(IN_2) \\ P_1(TN) &= P_1(A) \times P_0(IN_2), & P_0(TN) &= 1 - P_1(TN) \end{aligned} \tag{3}$$

where, P_0 denotes the success probability of a node and P_1 denotes the failure probability of a node.

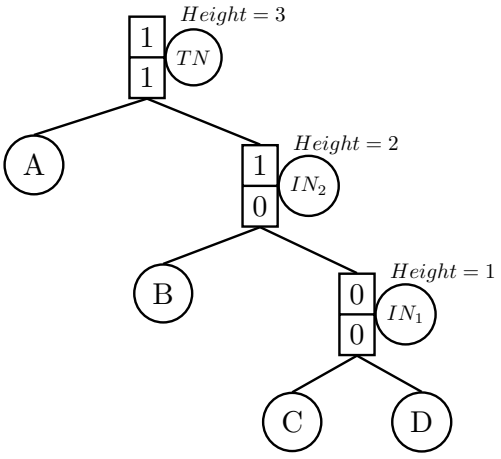


Figure 3. A simple Tree data structure with independent nodes

We propose a generic algorithm to calculate the probabilities of all the independent nodes in a Tree data structure. The required steps for obtaining the probabilities are described below:

1. Calculate the height of all internal nodes in the Tree data structure.
2. Sort the internal nodes in ascending order based on their height.
3. Calculate the binary state probabilities of the first internal node in the sorted list based on its logic gate connection and using Equation 2.
4. Repeat step 3 for all the internal nodes in the sorted list.
5. In the final step, calculate the binary state probabilities of the top node based on its logic gate connection and Equation 2.

TREE WITH TWO DEPENDENT CHAINS

The algorithm described in the previous section is only valid for a Tree data structure with independent nodes. When dependent or common nodes are present in a Tree data structure, the intersection probabilities of node states can no longer be obtained simply by multiplying their respective probabilities as shown in Equation 2. Therefore, in order to obtain the correct probability of the internal node, the intersection of its child nodes must be performed using Boolean algebra. In this study, we consider a simple Tree data structure (see Figure 4) with only one dependent node and a single loop to illustrate the process for calculating the exact probability of top node TN. The top node probability is calculated using Equation 4.

$$P_{Out}(TN) = \bigcap_{i=1}^{n_{chains}} P_{Inp}(Chain_i) \tag{4}$$

where, Out and Inp are the output and input states of the logic gate associated with the top node TN and $Chain_i$ is the i^{th} dependent chain connected to the top node.

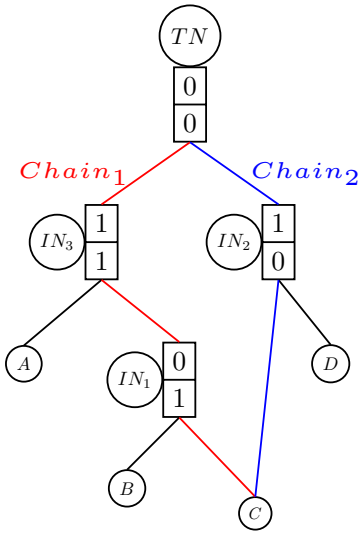


Figure 4. A simple Tree data structure with one dependent node and single loop

Equation 4 can be simplified and written as a generalized equation which is shown below.

$$P_0(TN) = K_1 + K_2 P(C) \qquad [K_1 = k_{1_1} k_{1_2}, K_2 = k_{1_1} k_{2_2} + k_{1_2} k_{2_1} + k_{2_1} k_{2_2}] \tag{5}$$

In this study, we propose a simple algorithm to calculate the coefficients k_{1_i} and k_{2_i} for a given chain i which can be used to calculate the coefficients K_1 and K_2 for the top node. k_{2_i} can be calculated by multiplying the probabilities of all the leaf nodes that are connected to the internal nodes along the chain i and is given by Equation 6.

$$k_{2_i} = (-1)^z \prod_{j=1}^{n_i} P_{InpIN}(N_j) \tag{6}$$

where, n_i is the total number of leaf nodes (except the common leaf node C) and other internal nodes connected to the internal nodes along the chain i. $P_{InpIN}(N_j)$ is the probability of the node N_j and the state of the node ($InpIN$) is determined based on its connection to the logic gate associated with the internal node. z is the total number of state changes between a child node output state to its parent node input state along the chain. Next, the term k_{1_i} is evaluated using Equation 7.

$$k_{1_i} = P_{Ind}(Chain_i) - k_{2_i} P(C) \tag{7}$$

where, $P_{Ind}(Chain_i)$ is the probability of chain i estimated using the algorithm for the Tree data structure with independent nodes.

COMPUTATIONAL EFFICIENCY

In this section, the computational efficiency of the proposed algorithm is compared to that of a traditional approach. To do so, relatively larger sized fault trees are considered and run times are calculated for both approaches. A set of 10 cases with 117 basic events, 15 intermediate events, and varying logic gate connections are considered. The fault trees are analyzed with a traditional approach that utilizes MOCUS and upper bound approach. The true computation run time or CPU time is highly dependent upon the type of hardware used. Therefore, for a hardware independent comparison, run times are normalized with respect to the run time for the proposed method on the same hardware. This assists with visualization of relative difference in the computation times using the two approaches. The normalized run times for each case in the three sets are compared in Figure 5. As seen in Figure 5, the run times of traditional approach can be significantly high in some cases depending upon the type and connectivity of different logic gates in the fault tree. An important aspect that is not directly evident in the figures is that the run time for proposed approach remains constant for various different cases of a given set. It changes from one set to another. In other words, unlike the traditional approach, the run time for the proposed approach is not dependent upon the type of logic gates and their connectivity within a given fault tree. Figure 6 compares the failure probability of top event for each case as evaluated from the two methods for the illustrating the accuracy of the proposed approach. It is also important to note that for all cases and each set, the failure probabilities of top node are identical from the two approaches as shown in Figure 6.

Figure 5. Comparison of efficiency of the proposed algorithm compared to traditional approach

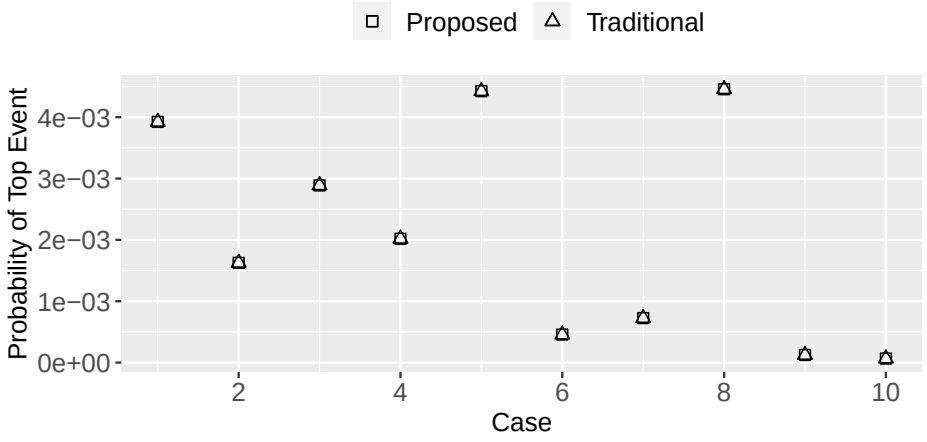


Figure 6. Comparison of accuracy of the proposed algorithm compared to traditional approach

SUMMARY AND CONCLUSIONS

Fault and event trees are used for probabilistic risk assessment of nuclear power plant systems. A fault and even trees analysis for a power plant requires modelling of hundreds of component failures, logic gates, multiple occurring events, and dependent events. Such interconnection for large networks can lead to excessive computational demand. Most of the traditional methods address computational demand by using assumptions or relying on high performance computing facilities that allow implementations of parallel computing. This study presents a novel module-based approach to address the computational demands of PRA. The proposed algorithm is developed for fault trees with a dependent node and multiple chains connecting the dependent node. The logic gates are converted into compressed truth tables to achieve the desired efficiency. For fault trees with fixed width and height (same number of basic events and intermediate events), the computational demand of traditional approaches changes based on different connectivity of events and gates. However, the computational demand of the proposed algorithm remains constant. In addition, the computational demand of proposed algorithm can be less than that of traditional approach by an order of magnitude for some cases.

In summary, the proposed approach is computationally efficient and has same accuracy as traditional approaches. However, it is different from traditional approaches in the sense that it does not calculate minimal cut sets. While the proposed algorithm is a significant improvement of the currently available techniques, it has not been applied to and explored for many different scenarios that exist in a real PRA network. The proposed algorithm needs to be enhanced for a few such scenarios such as consideration of Common Cause Failures (CCF) and n/m gates (Smith et al., 2011). Such improvements are recommended for future studies.

ACKNOWLEDGEMENTS

This research was supported by US Department of Energy (DOE) - Advanced Research Projects Agency - Energy (ARPA-E) under the grant DE-AR0000976. In addition, this research was partially supported by Center for Nuclear Energy Facilities and Structures at North Carolina State University. Resources for the Center come from the dues paid by member organizations and from the Civil, Construction, and Environmental Engineering Department and College of Engineering in the University.

References

Andersen, V. M., Lee, L. K., Patel, P. H. and Burns, E. T. (2013). "Seismic Probabilistic Risk Assessment Implementation Guide." Technical Report EPRI 3002000709, Electric Power Research Institute, Palo Alto, CA.

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

Bodda, S. S., Gupta, A. and Dinh, N. (2020a). "Enhancement of risk informed validation framework for external hazard scenario." *Reliability Engineering & System Safety*, 204, p. 107140.

Bodda, S. S., Gupta, A. and Dinh, N. (2020b). "Risk informed validation framework for external flooding scenario." *Nuclear Engineering and Design*, 356, p. 110377, ISSN 0029-5493, doi:https://doi.org/10.1016/j.nucengdes.2019.110377.

Bodda, S. S., Gupta, A., Ju, B. and Kwon, M. (2019). "Multi-hazard fragility assessment of a concrete floodwall." *Reliability Engineering and Resilience*, 1(2), pp. 46–66.

Bodda, S. S., Gupta, A., Ju, B. S. and Jung, W. (2020c). "Fragility of a Weir Structure due to Scouring." *Computational Engineering and Physical Modeling*, 3(1), pp. 1–15.

Bodda, S. S., Gupta, A. and Sewell, R. T. (2021). "Application of risk-informed validation framework to a flooding scenario." *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 7(4), p. 04021044.

Cavalieri, F., Franchin, P., Gehl, P. and D'Ayala, D. (2017). "Bayesian networks and infrastructure systems: Computational and methodological challenges." In *Risk and reliability analysis: theory and applications*, pp. 385–415, Springer.

Chandrakanth, B., Andrew, S., Justin, C. and Saran, B. (2019). "Special Session: Seismic Probablistic Risk Assessment Using MASTODON." In *Transactions of the 25th International Conference on Structural Mechanics in Reactor Technology*.

IAEA-Corporate-Author (2010). *Development and Application of Level 1 Probabilistic Safety Assessment for Nuclear Power Plants Specific Safety Guide*. International Atomic Energy Agency.

Jung, W. S. (2015). "A method to improve cutset probability calculation in probabilistic safety assessment of nuclear power plants." *Reliability Engineering & System Safety*, 134, pp. 134–142.

Kahle, D., Savitsky, T., Schnelle, S. and Cevher, V. (2008). "Junction tree algorithm." *Stat*, 631.

Kennedy, R. and Ravindra, M. (1984). "Seismic fragilities for nuclear power plant risk studies." *Nuclear Engineering and Design*, 79(1), pp. 47 – 68, ISSN 0029-5493, doi:https://doi.org/10.1016/0029-5493(84)90188-2.

Kumar, A., Kumar, A., Choudhary, S. and Varde, P. (2010). "Optimization of binary decision diagram using genetic algorithm." In *2010 2nd International Conference on Reliability, Safety and Hazard-Risk-Based Technologies and Physics-of-Failure Methods (ICRESH)*, pp. 168–175, IEEE.

RA-S, ASME (2002). "Standard for Probabilistic Risk Assessment for Nuclear Power Plant Applications, 2002."

Reed, S. (2017). "An efficient algorithm for exact computation of system and survival signatures using binary decision diagrams." *Reliability Engineering & System Safety*, 165, pp. 257–267.

Smith, C., Wood, S., Galyean, W., Schroeder, J. and Sattison, M. (2011). "Saphire 8 volume 2-technical reference (No. INL/EXT-09-17010)." Technical report, Idaho National Laboratory (INL).

Smith, C. L., Wood, T., Knudsen, J. and Ma, Z. (2016). "Overview of the SAPHIRE Probabilistic Risk Analysis Software." Technical report, Idaho National Lab.(INL), Idaho Falls, ID (United States).

US Nuclear Regulatory Commission (2002). "An approach for using probabilistic risk assessment in risk-informed decisions on plant-specific changes to the licensing basis." *Regulatory Guide, 1.174, Rev. 1*.

Vaishanav, P., Gupta, A. and Bodda, S. S. (2020). "Limitations of traditional tools for beyond design basis external hazard PRA." *Nuclear Engineering and Design*, 370, p. 110899.

Wegener, I. (2004). "BDDs—design, analysis, complexity, and applications." *Discrete Applied Mathematics*, 138(1-2), pp. 229–251.

Yeh, W.-C. (2006). "A new algorithm for generating minimal cut sets in k-out-of-n networks." *Reliability Engineering & System Safety*, 91(1), pp. 36–43.

Yeh, W.-C. (2021). "Novel binary-addition tree algorithm (BAT) for binary-state network reliability problem." *Reliability Engineering & System Safety*, 208, p. 107448.