An Investigation and Implementation of Quantification Methods for Probabilistic Contingency Analysis

Prepared for the U.S. DOE Office of Electricity

Advanced Grid Modeling (AGM) Program

Program Manager: Dr. Alireza Ghassemian

by

Meng Yue[1], Guilherme Larangeira[2], Feng Dong[3], and Robert Lofaro[1]

[1]Interdisciplinary Science Department, Brookhaven National Laboratory

[2]Department of Society and Technologies, Stony Brook University

[3]Siemens PTI

September 30, 2020

Table of Contents

Table of Figures:

Table of Tables:

Acronyms

BDD	Binary Decision Diagram
CMO	Common Mode Outage
IIO	Intermittency Induced Outages
ITE	If-Then-Else
MCUB	Minimum Cut set Upper Bound
MCS	Minimum Cut set
NPP	Nuclear Power Plant
PCA	Probabilistic Contingency Analysis
PRA	Probabilistic Risk Assessment
SDP	Sum of Disjoint Products

Executive Summary

This report summarizes the follow-up study for the project "Develop Risk-Informed Decision-Making Capability using Electric Power Industry Standard Planning Tools as Platform." In this extended study, BNL investigated and implemented a list of methods that can be used for better quantifying the reliability indexes in a probabilistic contingency analysis (PCA). These methods will be used with planning tools such as the Power System Simulator for Engineering (PSS/E) software tool used by power system planning engineers to simulate electrical power transmission networks. The implementation creates a standalone application that takes a list of contingencies that cause system problems and the outage rates and durations of the single outages generated by PSS/E and quantifies the system problem indexes using both approximation and exact methods.

This report first discusses the probability quantification methods, which are mainly borrowed from the practices in probabilistic risk assessments (PRA) that have been mandated and exercised for nuclear power plants (NPPs) for decades. In nuclear PRA, a fault tree/event tree approach [Vesely 1981, Russell 1993] is almost exclusively used. After processing event trees and fault trees, a set (usually a large set) of minimal cut sets are obtained and quantified to acquire the top event probability, as well as the associated uncertainties based on the distributions of the parameters for the basic events.

Assuming knowledge of the probabilities of individual outage events, quantification of the minimum cut sets is essentially the same as quantification of contingencies of different orders in a PCA. The BNL study focuses on existing quantification methods being used in nuclear PRA. It shows that while the rare event approximation has been frequently used in PCA, another method, called min cut upper bound (MCUB), enhances the quantification accuracy and is a better approximation. In addition, a binary decision diagram (BDD) approach [Bryant 1986] that can calculate the exact probabilities of the minimal cut sets has received much attention in recent years. The BDD diagram is briefly reviewed to understand its basics and is considered another potential candidate for PCA quantification.

Second, the report investigates the calculation of the probabilities, frequencies, and durations of component outage events in PCA studies. For the high order contingencies (double or triple contingencies), the discussion focuses on how to calculate the frequencies and duration times using the Markov model and the cut sets. An exact quantification scheme for the system problem reliability indexes based on the full Markov model is discussed. Alternatively, quantification of system problem reliability indexes based on the cut sets using either approximation methods, such as the rare event or MCUB approximation, or an exact method, i.e., BDD, is discussed. The major challenge is the need to include common mode failures or outages (CMOs). This challenge and how to address the challenge are also discussed in detail.

Finally, a Python implementation of the investigated PCA quantification methods was performed and a comparison study of PCA system problem probabilities calculated using different methods was conducted to show that the developed quantification module can be a standalone application and part of the enhanced PCA tool developed in [Yue 2019]. The Python quantification module takes a list of contingencies generated by PSS/E that cause system problems and the outage rates and durations of single outages, and quantifies the probability, frequency, and duration of the system problems.

In a summary, the major contributions of this study are:

(1) Identification of the quantification issues with the existing methods being used in PCA software tools;
(2) Introduction of the MCUB for improved approximation and Markov model and BDD method for exact calculation of system problem probabilities;
(3) Extension and development of the Markov model and BDD methods for quantifying the system problem frequencies and duration;
(4) Investigation of the Markov model approach for the PCA quantification;
(5) Implementation of the proposed MCUB and BDD methods in a Python package, which can be a standalone package or incorporated in the ePCA tool that was developed and delivered in the original project.

A comparison study was performed, which shows that:

(1) Both rare event and MCUB produce conservative results in terms of total probability and duration time while the BDD method can provide exact solutions;
(2) The results using MCUB, although still conservative, appear much less conservative than the results from the rare event approximation;
(3) The BDD method is more time consuming than the rare event and MCUB approximation but appear manageable and can be a feasible approach for the PCA quantification.

The Python-based PCA quantification package interfaces with the contingency analysis tool by taking the list of contingencies contributing to the system problems and outage statistics to quantify the system problem indexes and will significantly increase the capabilities of the existing PCA tools.

The enhanced PCA or ePCA tool has been developed with state-of-the-art capabilities by developing and incorporating a new modeling method for renewable generation, which enables inclusion of solar generation in the PCA by considering its correlation with wind. Future work may include the development of a user-friendly interface and post-processor to create load duration points from each case studied, then interpolate and/or extrapolate among these points to have an approximated load duration curve that can be overlaid by the calculated unserved energy using the same approach. In addition, a case study can be performed using the latest ePCA tool together with NERC guidelines and renewable generation data by considering both solar and wind generation in a real utility system.

1. Introduction

In July 2019, BNL completed the original scope of work for the DOE/AGM project titled "Develop Risk-Informed Decision-Making Capability using Electric Power Industry Standard Planning Tools as Platform." In this study, a probabilistic contingency analysis (PCA) framework was developed for addressing intermittency induced outages (IIOs) and common mode IIOs associated with wind generation. The probabilistic contingency analysis framework developed is a Python-driven program that can be used with the Power System Simulator for Engineering (PSS/E) software tool used by power system engineers to simulate electrical power transmission networks. A number of case studies were performed and the final deliverables were completed [Yue 2019].

The original study provided a number of important insights related to probabilistic contingency analyses, including the following:

- The study showed that data poolability is indeed an issue that needs to be resolved first before performing a PCA. As shown in this study, outage data for many different types of grid components cannot be pooled and their statistics need to be described by distributions.
- The results also showed that a majority of outage data from conventional generators can be pooled. For the rest of conventional generators, the distributions of their outage statistics are relatively narrow compared to other grid components. These results also are an indicator that environmental conditions have significant impacts on grid component outages.
- Based on the analyses of actual renewable generation data, the results showed that generation of a single wind farm can ramp up and down within a very short time period (i.e., tens of minutes). In addition, this type of ramping event can happen concurrently for multiple wind farms and can cause more significant impacts on grid operation. The impacts can be similar to conventional generator outages, and such fast ramps should be considered as outages.
- Case studies performed in this project showed that Monte Carlo simulations implemented in the enhanced PCA are capable of computing the true mean values and providing statistics of system problem indices. This will provide valuable information for developing mitigation actions in the planning process.
- Increasing penetration levels of wind generation would definitely have increasingly larger, but different impacts on grid reliability. Due to the nature of outages related to intermittency of renewable generation, the system problem frequencies would increase and the durations will decrease.

As a follow-up to the original project scope, an additional study was performed to further refine the enhanced PCA tool. The PCA built into existing software, including PSS/E, calculates probabilistic indices of system problems approximately based on a rare event approximation. When the probabilities of contingencies are relatively large, the rare event approximation can no longer be used, and calculated probabilities may deviate significantly from the real values, or even be larger than 1.0, which was observed in the studies performed in [Yue 2019]. This is especially true for higher order contingencies since they are not mutually exclusive. Theoretically, the total probability can only be calculated exactly using the inclusion-exclusion principle, which may be difficult because large computational efforts are required when the number of contingencies is large. A new quantification scheme is needed and was developed to more precisely calculate the probabilistic indices in the PCA.

This report summarizes the results of the follow-up study for the project "Develop Risk-Informed Decision-Making Capability using Electric Power Industry Standard Planning Tools as Platform." In this extended study, BNL investigated and implemented a list of methods that can be used for better quantifying the reliability indexes in the PCA. The implementation creates a standalone application that takes a list of contingencies that cause system problems and the outage rates and durations of the single outages generated by PSS/E and quantifies the system problem indexes using both approximation and exact methods.

This report first discusses the probability quantification methods, which are mainly borrowed from the practices in probabilistic risk assessment (PRA) that have been mandated and exercised for nuclear power plants (NPPs) for decades. In nuclear PRA, a fault tree/event tree approach [Vesely 1981, Russell 1993] is almost exclusively used. After processing event trees and fault trees, a set (usually a large set) of minimal cut sets[1] are obtained and quantified to acquire the top event[2] probability, as well as the associated uncertainties based on the distributions of the parameters for the basic events.

Assuming knowledge of the probabilities of individual outage events, quantification of the minimum cut sets is essentially the same as quantification of contingencies of different orders in PCA. The BNL study focuses on existing quantification methods being used in nuclear PRA. It shows that while the rare event approximation has been frequently used in PCA, another method, called min cut upper bound (MCUB), enhances the quantification accuracy and is a better approximation. In addition, a binary decision diagram (BDD) approach [Bryant 1986] that can calculate the exact probabilities of the minimal cut sets has received much attention in recent years. The BDD diagram is briefly reviewed to understand its basics and is considered another potential candidate for PCA quantification.

Second, the report investigates the calculation of the probabilities, frequencies, and durations of component outage events in PCA studies. For high order contingencies (double or triple contingencies), the discussion focuses on how to calculate the frequencies and duration times using the Markov model and the cut sets. An exact quantification scheme of the system problem reliability indexes based on the full Markov model is discussed. Alternatively, the quantification of system problem reliability indexes can also be performed based on the cut sets using either approximation methods such as rare event or MCUB approximation or exact method, i.e., BDD, is discussed presented. The major challenge is the need to include common mode failures or outages (CMOs). This challenge and how to address the challenge are also discussed in detail.

Finally, a Python implementation of the investigated PCA quantification methods was performed and a comparison study of PCA system problem probabilities calculated using different methods was conducted to show that the developed quantification module can be a standalone application and part of the enhanced PCA tool developed in [Yue 2019]. The Python quantification module takes a list of contingencies that causes system problems, which are generated by PSS/E, and the outage rates and

[1] A minimal cut set is the smallest combination (sufficient) of component failures or basic events which, if they all occur, will cause the top event to occur.

[2] In nuclear PRA, a top event indicates an undesirable event such as reactor core damage. In contingency analysis for power grid, the counterpart of a top event can be the loss of load or a voltage violation issue, and the cut sets are the contingencies that cause such a loss of load or voltage issue. Therefore, in this report, top event and system problem and cut sets and contingencies will be used interchangeably.

duration of single outages, and quantifies the probability, frequency, and duration of the system problems.

2. Survey of Quantification Methods for Contingencies

2.1 Rare Event Approximation

The Sylvester-Poincare development [Russell 1993] (or inclusion-exclusion principle) computes the exact probability of a union of basic events $\{E_i, i = 1,2,\cdots,K\}$:

$$P(E_1 \cup E_2 \cup \cdots \cup E_K) = \sum_{1\le i\le K} P(E_i) - \sum_{1\le i<j\le K} P(E_i \cap E_j) + \sum_{1\le i<j<k\le K} P(E_i \cap E_j \cap E_k) - \cdots + (-1)^K P(E_i \cap \cdots \cap E_K) \quad (1)$$

This is well-known but can be intractable for a large number of events. One commonly used method to deal with this issue is to use the rare event approximation that ignores the simultaneous occurrences of two or more rare events. The rare event approximation is actually the first term of Equation (1), i.e.,

$$P(E_1 \cup E_2 \cup \cdots \cup E_K) = \sum_{1\le i\le K} P(E_i) \quad (2)$$

It is noted that the rare event approximation in Equation (2) is the upper bound of the probability while the lower bound is given by the first two terms, i.e.,

$$P(E_1 \cup E_2 \cup \cdots \cup E_K) = \sum_{1\le i\le K} P(E_i) - \sum_{1\le i<j\le K} P(E_i \cap E_j)$$

which is also called the Boole-Bonferroni bound [Russell 1993]. Such a rare event approximation can be very conservative, especially when the probabilities of the basic events are relatively large (e.g., $> 10^{-1}$) and/or the number of basic events is very large. Sometimes, the total probability calculated using the rare event approximation can be larger than 1.0 [Epstein 2005].

The Sylvester-Poincare development can also be applied to the probability of a union of cut sets by replacing the basic events with cut sets. In nuclear PRA, the quantification of minimal cut sets also relies on a rare event approximation due to the huge number of cut sets [Russell 1993]. Assume that a Boolean formula for a top event Top is represented by the sum of a set of minimal cut sets, i.e.,

$$Top = \sum_{\pi\in MCS[Top]} \pi$$

The probability of the top event $P(Top)$ is:

$$P(Top) = \sum_{\pi\in MCS[Top]} P(\pi) \quad (3)$$

where $MCS[Top]$ indicates the cut sets of the top event Top. Each of the minimal cut sets may consist of a combination of component failures, e.g., a generator outage and a transmission line outage. Replacing basic events E_i in Equations (2) and (3) will also lead to the upper bound and lower bound of the top event. Again, the rare event approximation can generate very conservative results here.

In PCA, the probability quantification of a system reliability index can be represented by Equation (3) in the same way. The top event is actually one of the system problems, such as loss of load, voltage violation, or overloading, while a cut set π is a single order or high order contingency, such as a combination of Generator A trip and Line C outage. The set $MCS[Top]$ indicates the collection of such contingencies that caused the specific system problem. Each of the single order outages is characterized by its outage rate and duration time, i.e., the time to repair. Therefore, solving for the probabilities of system problems in

PCA is essentially the same as for the top events. Actually, the rare event approximation method is the most commonly, if not exclusively, used in PCA of the existing software tool. Other top event quantification methods will be investigated below to address the conservativeness of the rare event approach.

2.2 Minimal Cut set Upper Bound

Calculating the minimal cut set upper bound (MCUB) [Esary 1970] is another approximation that is better than the rare event approximation in Equation (2). MCUB, also a first order approximation, has been often used in practice and the total probability will never exceed 1.0 [Russell 1993], unlike the rare event approximation. Since the top event is the sum (Boolean) of the minimal cut sets,

$$P(Top) = P(at\ least\ one\ of\ the\ minimal\ cutset\ occurs)$$

$$= 1 - P(no\ minimal\ cutset\ occurs)$$

It is also true that

$$P(no\ minimal\ cutset\ occurs) \geq \prod_{i=1}^{K} P(minimal\ cutset\ i\ does\ not\ occur) \qquad (4)$$

where K is the total number of cut sets for the top event. Equality in Equation (4) holds only when Equation (1) events are independent and no event occurs in more than one cut set. Therefore, for $P(Top) = \sum_{\pi \in MCS[Top]} P(\pi) = \sum_{i=1}^{K} P(\pi_i)$, the MCUB is given by

$$P(Top) = 1 - \prod_{i=1}^{K} (1 - P(\pi_i)) \qquad (5)$$

If the equality in Equation (4) holds, then Equation (5) gives the exact top event probability without any approximation, which is usually not the case in practice. In a nuclear PRA, the MCUB works well for quantification of fault trees of coherent systems, i.e., with only AND and OR gates and without complemented events or NOT gates. Generally,

$$P(Top) \leq minimal\ cutset\ upper\ bound \leq rare\ event\ upper\ bound$$

is true. The simplicity and less conservativeness of MCUB approximation makes it an attractive candidate for the PCA quantification.

2.3 Cut set Probability Truncation

This is almost always used in nuclear PRA and, indeed, is very straightforward, i.e., the cut set of a probability lower than a pre-defined threshold will be excluded from the top event probability quantification [Vesely 1981, Russell 1993]. The rationale behind this is that the probabilities of some cut sets are too small to have a significant impact on the final result. However, one has to be aware that the number of the omitted cut sets can be huge, and the total contribution of these omitted cut sets can be fairly large. The selection of the threshold is a trade-off between the tolerance and the computational resources.

2.4 Binary Decision Diagram

2.4.1 Shannon Decomposition and ITE Connectives

While the above approximation methods make it practical for quantifying the complicated event tree and fault tree in PRA, they also overestimate the results in many cases [Epstein 2005]. An alternative cut set quantification method is the so-called binary decision diagram or BDD [Bryant 1986]. The BDD of a Boolean formula is a compact encoding of the truth table of the formula and, therefore, can provide the ***exact*** values of the probabilistic measures without any approximation. The BDD development is based on the Shannon Decomposition [Bryant 1986, Rauzy 1997]

$$f(x_1, x_2, \cdots, x_n) = x_1 \cap f(1, x_2, \cdots, x_n) \cup \bar{x}_1 \cap f(0, x_2, \cdots, x_n) \quad (6)$$

Or simply as

$$f(x_1, x_2, \cdots, x_n) = x_1 f(1, x_2, \cdots, x_n) + \bar{x}_1 f(0, x_2, \cdots, x_n) \quad (7)$$

where $x_i, i = 1,2, \ldots, n$, is a Boolean variable and f is a Boolean function. $\bar{x}$ means "NOT x". Note that the two terms in the right side of either Equation (6) or Equation (7) are mutually exclusive.

Any Boolean operators or connectives (e.g., AND, OR, XOR) can be represented by a so-called ITE connective or if-then-else connective [Rauzy 1993], which is by definition:

$$ite(f, g, h) = f \cap g \cup \bar{f} \cap h$$

$$= fg + \bar{f}h$$

The ITE connective means that, if the first argument holds, then return the second argument; otherwise, the third argument.

The Shannon decomposition in Equation (6) can be represented by the ITE connectives as:

$$f(x_1, x_2, \cdots, x_n) = ite(x_1, f(1, x_2, \cdots, x_n), f(0, x_2, \cdots, x_n))$$

It can be easily seen that a top event $f = x$ can be represented by $ite(x, 1,0) = x \cdot 1 + \bar{x} \cdot 0$, which means that if $x = true$ $(x = 1)$, then $f = 1$. Otherwise, $f = 0$. I.e.,

$$f = x = ite(x, 1,0) \quad (8)$$

Any basic events can be represented by the ITE connective, as shown in Equation (8). The Shannon decomposition lays the foundation for the BDD.

2.4.2 Graphic Representation of BDDs

Figure 1 shows the structure of a graphical representation of the BDD for simple function $f = x$ in Equation (8). x is the only variable and also the top (root) node or the vertex. Nodes "1" and "0" are leaf nodes. The leg to the **then** branch (or 1-branch) in Figure 1 is called the AND-leg while the one to the **else** branch (or 0-branch) is called the OR-leg [Way 2000], as suggested by the ITE structure.

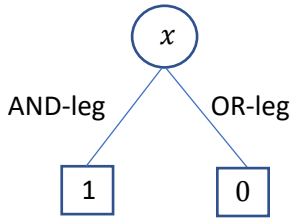

Figure 1: The BDD structure of $f = x$.

For an event $Top = x_1 x_2$, a possible ITE connective representation can be $Top = ite(x_1, ite(x_2, 1,0), 0)$ by taking the order $x_1 < x_2$, where $x_1 < x_2$ means that node x_1 is closer to the root node than x_2. A BDD structure for the ITE connective representation of this top event is shown in Figure 2.

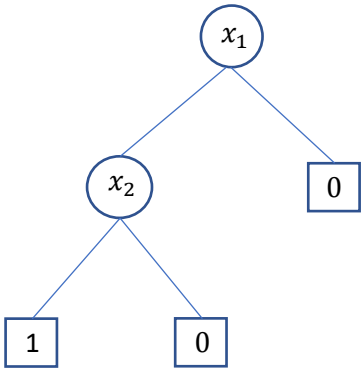

Figure 2: A possible BDD structure of an AND gate.

And for an event $Top = x_1 + x_2$, a possible ITE connective representation can be $Top = ite(x_1, 1, ite(x_2, 1,0))$ and the corresponding BDD structure can be seen in Figure 3.

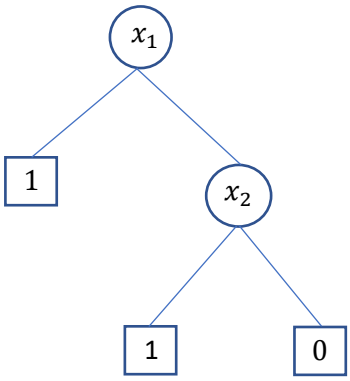

Figure 3: A possible BDD structure of an OR gate.

The examples in Figures 2 and 3 provide an intuitive understanding of why the legs to the left and right in a BDD are called AND-leg and OR-leg, as shown in Figure 1.

By using the notation of AND leg and OR leg in Figure 1, [Way 2000] proposed a method for developing BDDs. If the logical connective between variables x_1 and x_2 is AND, then x_2 can be simply connected to the left leg (AND-leg or 1-branch) of x_1. Otherwise, x_2 is connected to the right leg (OR-leg or 0-branch) of x_1. This can be easily verified by inspecting Figures 2 and 3. In Figure 2, only when x_1 and x_2 are both 1 (true), then the top event is 1. In Figure 3, if x_1 or x_2 is 1, then the top event is 1. This can be extended to another more complicated example. For example, a top event $Top = a(b + c + de)$ can be easily translated into the BDD structure [Way 2000] shown in Figure 4:

Such a BDD construction method is more applicable for logical operations with only a single descendent [Way 2000]. In practice, the construction of a BDD is usually done by applying the Shannon Decomposition principle in Equation (8) recursively based on operation of ITE connectives [Rauzy 1993, Sinnamon 1996], i.e.,

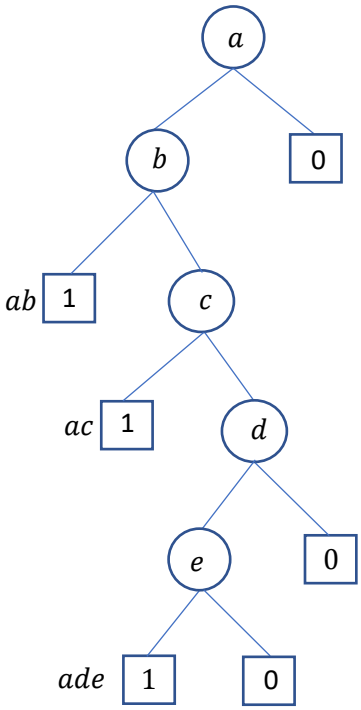

Figure 4: A possible BDD structure of $Top = a(b + c + de)$.

For $x \neq y$, let $J = ite(x, G_1, H_1)$ and $K = ite(y, G_2, H_2)$, then:

$$J < op > K = ite(x, G_1 < op > K, H_1 < op > K) \quad (9)$$

For $x = y$, let $J = ite(x, G_1, H_1)$ and $K = ite(x, G_2, H_2)$, then:

$$J < op > K = ite(x_1, G_1 < op > G_2, H_1 < op > H_2) \quad (10)$$

where $< op >$ represents any Boolean operations of the logic connectives such as AND and OR.

By doing so, one can rewrite any Boolean function as an equivalent one that is built only with variables, connectives ITE, and Boolean constants 0 and 1.

Note that there can be multiple ITE connective representations for a single formula, e.g., if the order of variables is changed, then different ITE representations can be obtained.

2.4.3 Qualitative and Quantitative BDD Analyses

In terms of qualitative and quantitative analyses of the top event, i.e., identification and quantification of the cut sets, it is also very straightforward after acquiring the BDD representation of a top event. Since each path from the top or vertex of the BDD defines a solution to the Boolean representation, the cut set of the fault tree can be obtained by tracing from the top node to a leaf node "1". The paths include only the basic events that lie on the "1" branch on the way to a leaf node "1", i.e., ab and ac in Figure 4.

For the top event $Top = a(b + c + de)$, it is easy to tell that the cut sets are ab, ac, ade. If we trace to the leaf nodes "1" of the top event BDD representation in Figure 4, we will see that the corresponding paths are: ab, ac, and ade, as indicated in the figure. Note that the path means that all the associated events need to occur for the top event to occur, i.e., a path is a cut set. For example, the path ade means that a, d, and e all need to occur to lead to the top event. However, the paths in the BDD are disjoint and the sum of probabilities of the paths is the exact probability of the top event.

In the calculation of the cut set probability, the basic events that lie on the 0 branch or OR-leg need to be included to count the probabilities that these event do not happen, i.e., $P(Top) = P\big(ab + a\bar{b}c + a\bar{b}\bar{c}de\big) = P(ab) + P\big(a\bar{b}c\big) + P\big(a\bar{b}\bar{c}de\big)$, i.e.,

$$P(Top) = P(a)P(b) + P(a)\big(1 - P(b)\big)P(c) + P(a)\big(1 - P(b)\big)\big(1 - P(c)\big)P(d)P(e)$$

This is the exact probability of the top event and can be confirmed easily using the Sylvester-Poincare development in Equation (1).

A BDD is generally difficult to be applied to a large-scale fault tree/event tree quantification because the size of the BDD increases exponentially with the number of variables. However, it is less difficult to be applied to coherent fault tree, i.e., (a) its structure function (Boolean function) is increasing (non-decreasing) and (b) each basic event is relevant. A fault tree with only AND and OR connectives will be coherent. A fast algorithm was developed for analyzing coherent fault trees in [Jung 2004]. In addition, the size and optimality of BDDs are closely related to the ordering of the basic events when creating BDDs. One way of dealing with this is to generate BDDs using the repeated basic events first, as discussed in [Sinnamon 1996].

Considering the issue with the rare event approximation and the fact that only the AND and OR logical operations are involved in the PCA quantification, the MCUB approach can be a better method, i.e., by replacing the cut sets in Equation (4) with the individual contingencies of different orders.

For an example cut set or a sum of the contingencies, $Top = x_1x_2 + x_2x_3 + x_1x_3$, if we assume that each basic event or contingency has the same failure probability p, then the rare event approximation of the top event gives that

$$P(Top) = 3p^2$$

while the MCUB of the top event is

$$P(Top) = 1 - (1-p^2)(1-p^2)(1-p^2)$$
$$= 3p^2 - 3p^4 + p^6$$

The exact probability of the top event can be calculated as

$$P(Top) = P(x_1x_2) + P(x_2x_3) + P(x_1x_3) - P(x_1x_2 \cap x_2x_3) - P(x_1x_2 \cap x_1x_3) - P(x_2x_3 \cap x_1x_3) + P(x_1x_2 \cap x_2x_2 \cap x_1x_3)$$
$$= 3p^2 - 2p^3$$

by using the Sylvester-Poincare development in Equation (1). Therefore, the MCUB result is obviously less conservative than the rare event approximation.

In the PCA application, x_1, x_2, and x_3 can be outages of individual component outages, e.g., Line 1, Line 2, and Line 3 outages. The top event can be the system problems, i.e., over-/under-voltage, overloading, a loss of load. Therefore, the physical meaning of the given top event can be an overvoltage issue caused by the outages of any two of the three lines, Lines 1, 2, and 3.

The BDD approach can also be used to calculate the probability of the same example top event here. The top event can be rewritten as $Top = x_1(x_2 + x_3) + x_2x_3$, which can be represented by the ITE connectives, if we use the order $x_1 < x_2 < x_3$, as:

$$Top = ite(x_1, 1, 0)\big(ite(x_2, 1, 0) + ite(x_3, 1, 0)\big) + ite(x_2, 1, 0)ite(x_3, 1, 0)$$
$$= ite(x_1, 1, 0)ite\big(x_2, 1, ite(x_3, 1, 0)\big) + ite(x_2, ite(x_3, 1, 0), 0)$$
$$= ite(x_1, ite(x_2, 1, ite(x_3, 1, 0)), 0) + ite(x_2, ite(x_3, 1, 0), 0)$$
$$= ite(x_1, ite\big(x_2, 1, ite(x_3, 1, 0)\big) + ite(x_2, ite(x_3, 1, 0), 0), ite(x_2, ite(x_3, 1, 0), 0))$$
$$= ite(x_1, ite(x_2, 1, ite(x_3, 1, 0)), ite(x_2, ite(x_3, 1, 0), 0)) \quad (11)$$

In the above calculation, the operation rules of ITE connectives shown in Equations (9) and (10) are repeatedly used. From Equation (11), the BDD representation of the top event can thus be shown in Figure 5:

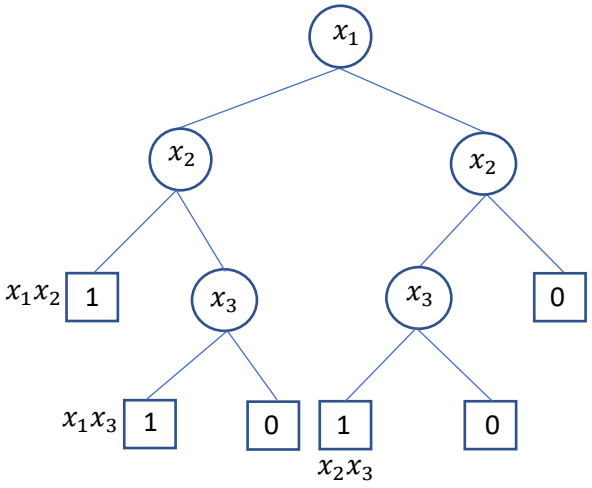

Figure 5: A BDD structure of $Top = x_1x_2 + x_2x_3 + x_1x_3$.

Figure 5 shows that the cut sets are x_1x_2, x_1x_3, and x_2x_3 by enumerating the paths from the top node to "1" leaf nodes. The probability of the top event is calculated by counting the probabilities of the events that do and do not occur along the paths, i.e.,

$$\begin{aligned} P(Top) &= P(x_1x_2) + P(x_1\bar{x}_2x_3) + P(\bar{x}_1x_2x_3) \\ &= p^2 + p^2(1-p) + p^2(1-p) \\ &= 3p^2 - 2p^3 \end{aligned}$$

which is apparently the exact solution of the top event without any approximation.

This simple case here is given only to illustrate the basic idea behind the rare event, MCUB, and BDD. Computer code will be needed to deal with a large number of cut sets or contingencies that need to be quantified. In addition, only the probability calculation is discussed. In PCA, system problem frequencies and durations are also of interest. These issues will be further investigated below.

3. System Reliability Indexes Calculation in PCA

3.1 Calculation of System Problem Probability in PCA

Compared to the fault tree/event tree applications in a nuclear PRA, the quantification in a PCA is even more straightforward since only basic logical operations AND and OR are involved and there is no need to quantify the success branches that include negations [Epstein 2005]. In the PCA, the probabilities of individual contingencies including single, double, triple, or even higher order contingencies that cause system problems, i.e., voltage, overloading, or loss of load, need to be calculated separately and summed to calculate the total probability of causing system problems.

From this perspective, the process is exactly the same as the quantification of the minimum cut sets in a nuclear PRA. Currently, a rare event approximation is the most commonly used quantification method for PCA applications. The truncation process is not very often used since the major barrier to a PCA is the difficulty performing a contingency analysis, not the number of cut sets or contingencies, i.e., high order contingencies makes it difficult to solve power flow and often lead to the divergency of power flow. Usually, triple contingencies are only considered in very long-term planning activities. Therefore, truncation process may be considered automatically in PCA.

3.2 Calculation of Problem Frequencies and Durations in PCA

The previous sections focused on quantification of the top event probabilities for the given collection of cut sets (or contingencies) assuming that the probabilities of individual component outages are known. In the PCA applications, the top event or system problem probabilities are only one type of the reliability indexes that need to be calculated. Other indexes include system problem frequencies and duration times, which will be further investigated in this section.

This section first investigates the calculation of the probabilities, frequencies, and durations of individual component outages or single contingencies based on the state transitions in the Markov model. For the high order contingencies (double or triple contingencies), the discussion focuses on how to calculate the frequencies and duration times using the Markov model. The procedures for calculating the system problem reliability indexes are presented. An exact quantification scheme of the system problem reliability indexes based on the full Markov model is also discussed, although difficult to implement in practice, especially in the presence of common mode failures or outages (CMOs).

3.2.1 Single Grid Component Outages

In existing probabilistic contingency analysis, almost every grid component outage is described using a bi-state model, i.e., in-service (or up) and out-service (or down). A simple Markov model can be used to describe the component A outage, as shown in Figure 6;

Figure 6: A Two-state Markov Model for Component A Outage.

The probabilities of Component A at Up and Dn states[3] can be solved as:

$$P_{AUp}(t) = \frac{\mu_A}{\lambda_A + \mu_A} + \left(1 - \frac{\mu_A}{\lambda_A + \mu_A}\right) e^{-(\lambda_A + \mu_A)t}$$

$$= \frac{\mu_A}{\lambda_A + \mu_A} + \frac{\lambda_A}{\lambda_A + \mu_A} e^{-(\lambda_A + \mu_A)t}$$

$$P_{ADn}(t) = \frac{\lambda_A}{\lambda_A + \mu_A}(1 - e^{-(\lambda_A + \mu_A)t}) \quad (12)$$

Therefore, the stead state probabilities of the two states for Component A at Up and Dn states are:

$$P_{AUp} = \frac{\mu_A}{\lambda_A + \mu_A}, \quad P_{ADn} = \frac{\lambda_A}{\lambda_A + \mu_A}$$

where λ_A is the failure rate or outage rate of Component A and μ_A is the repair rate, or $\mu_A = \frac{1}{T_{ADn_{Duration}}}$ with $T_{ADn_{Duration}}$ the duration time of Component A outage. Apparently, $P_{AUp}(t) + P_{ADn}(t) = 1.0$.

More generically, if a system with multiple states is observed for T hours, T_i hours are spent in state i, and number of transitions from state i to j is n_{ij} during the period of observation, then the transition rate from state i to j is given by $\lambda_{ij} = \frac{n_{ij}}{T_i}$, and the steady-state probability of being at state i is $P_i = \frac{T_i}{T}$.

The concept of frequency $F_{i,j}$ of encountering state j from state i is the expected (mean) number of transitions from state i to state j per unit time, i.e.,

$$F_{i,j} = \frac{n_{ij}}{T} = \frac{n_{ij}/T_i}{T/T_i} = \lambda_{ij} P_i \quad (13)$$

In the state transitions, ***the rule of frequency balance*** always holds, i.e., frequency of encountering a state (or a subset of states) equals to the frequency of exiting from the state (or the subset of states). Therefore, for the two-state component A outage model in Figure 6, the frequencies of encountering state Up and state Dn should be the same, i.e., $F_{Up,Dn}(t) = F_{Dn,Up}(t)$. To simplify the notation, we let $F_{Up,Dn}(t) = F_{ADn}(t)$ and can be represented by $F(t)$. This together with Equation (13), we have

$$F(t) = P_{AUp}(t)\lambda_A = P_{ADn}(t)\mu_A$$

Since $\mu_A = \frac{1}{T_{ADn_{Duration}}}$, we have

$$F(t = T) = \frac{P_{AUp}(t=T)}{T_{ADn_{Duration}}} \quad (14)$$

in the observation time T. Equation (12) and (14) are used to calculate the probability and the frequency of Component A outage with the known outage duration time $T_{ADn_{Duration}}$.

[3] Sometimes the probability of the down state can be approximated by $P_{A_{Dn}}(t) = \lambda_A t$. This needs to be used cautiously since repairs may not be accounted for here.

3.2.2 Higher Order Component Outages

Similarly, for a double outage model of Component A and Component B, the Markov state transition diagram is shown in Figure 7 below.

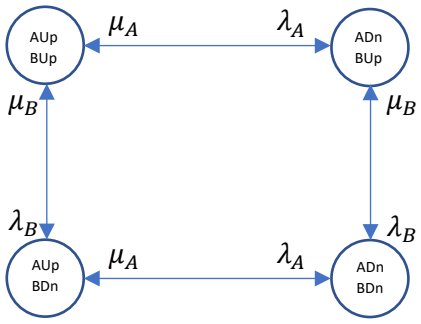

Figure 7: A Markov Model for Component A and B Outages.

Denoting $P_1 = P_{AUp,BUp}$, $P_2 = P_{ADn,BUp}$, $P_3 = P_{AUp,BDn}$, and $P_4 = P_{ADn,BDn}$, then we have

$$\frac{d}{dt}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} = \begin{bmatrix} -(\lambda_A+\lambda_B) & \mu_A & \mu_B & 0 \\ \lambda_A & -(\lambda_A+\mu_A) & 0 & \mu_B \\ \lambda_B & 0 & -(\lambda_A+\mu_B) & \mu_A \\ 0 & \lambda_B & \lambda_A & -(\mu_B+\mu_B) \end{bmatrix}\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

The time domain solutions of the states[4] are given as:

$$\begin{aligned} P_4(t) &= P_{ADn,BDn} \\ &= \frac{\lambda_A\lambda_B}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)}\left(1 - e^{-(\lambda_A+\mu_A)t} - e^{-(\lambda_B+\mu_B)t} + e^{-(\lambda_A+\mu_{A+}+\lambda_B+\mu_B)t}\right. \\ &= \frac{\lambda_A}{(\lambda_A+\mu_A)}\left(1 - e^{-(\lambda_A+\mu_A)t}\right) \times \frac{\lambda_B}{(\lambda_B+\mu_B)}\left(1 - e^{-(\lambda_B+\mu_B)t}\right) \\ &= P_{ADn}(t)P_{BDn}(t) \end{aligned} \quad (15)$$

i.e., the probability of both components being down is the product of the probabilities of individual components being down. In fact, for other states in Figure 7, we also have

$$\begin{aligned} P_1 &= P_{AUp,BUp}(t) = P_{AUp}(t)P_{BUp}(t), \\ P_2 &= P_{ADn,BUp}(t) = P_{ADn}(t)P_{BUp}(t), \\ P_3 &= P_{AUp,BDn}(t) = P_{AUp}(t)P_{BDn}(t) \end{aligned} \quad (16)$$

Therefore, the steady-state probabilities of all states can be easily calculated, i.e.,

[4] Sometimes, the probability of both A and B being failed can be approximated by using $P_{ADn,BDn} = (\lambda_A t)(\lambda_B t) = \lambda_A\lambda_B t^2$, which again needs to be used cautiously.

$$P_1 = \frac{\mu_A\mu_B}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)} \qquad P_2 = \frac{\lambda_A\mu_B}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)}$$
$$P_3 = \frac{\lambda_B\mu_A}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)} \qquad P_4 = \frac{\lambda_A\lambda_B}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)}$$

Note the probability of both component down $P_{ADn,BDn}$ in Equation (15) is the product of the probabilities of Component A being down and Component B being down, each can be calculated using Equation (12). Following Equation (13), because the total rate exiting state (ADn, BDn) is $(\mu_A + \mu_B)$, the frequency of encountering state (ADn, BDn) is denoted as $F(t)$ and can be calculated as:

$$F(t) = P_4(t)(\mu_A + \mu_B)$$
$$= P_4(t)\left(\frac{1}{T_{ADn_{Duration}}} + \frac{1}{T_{BDn_{Duration}}}\right) \quad (17)$$

$$T = T_{ADn,BDn} = \frac{1}{\mu_A+\mu_B} \quad (18)$$

The probability, frequency, and duration time of both Components A and B being down are calculated using Equations (15), (17), and (18) in the given time period T, e.g., one year.

The same strategy can be used to calculate the probability, frequency, and duration time of a triple component outage. The Markov model for three bi-state components A,B, and C can be found in many references and an example with different failure rates and repair rates defined for each component is shown in Figure 8 [Sperandio 2006]:

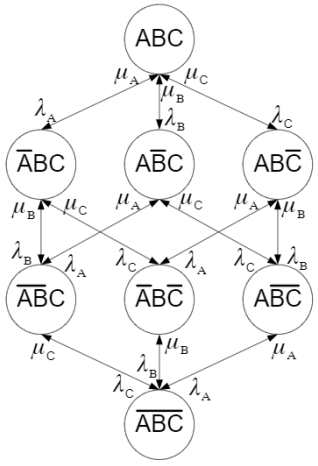

Figure 8: A Markov Model for Component A, B, and C Outages.

where $\bar{A}$, $\bar{B}$, and $\bar{C}$ indicate the down status of the components, i.e., $P_{ADn,BDn,CDn}(t) = P_{\bar{A},\bar{B},\bar{C}}(t)$.

$$P_{ADn,BDn,CDn}(t) = \frac{\lambda_A\lambda_B\lambda_C}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)(\lambda_C+\mu_C)}(1 - e^{-(\lambda_A+\mu_A)t} - e^{-(\lambda_B+\mu_B)t} - e^{-(\lambda_C+\mu_C)t} +$$
$$e^{-(\lambda_A+\mu_A+\lambda_B+\mu_B)t} + e^{-(\lambda_A+\mu_A+\lambda_C+\mu_C)t} + e^{-(\lambda_B+\mu_B+\lambda_C+\mu_C)t} -$$
$$e^{-(\lambda_A+\mu_A+\lambda_B+\mu_B+\lambda_C+\mu_C)t})$$
$$= \frac{\lambda_A}{(\lambda_A+\mu_A)}\left(1 - e^{-(\lambda_A+\mu_A)t}\right) \times \frac{\lambda_B}{(\lambda_B+\mu_B)}\left(1 - e^{-(\lambda_B+\mu_B)t}\right) \times \frac{\lambda_C}{(\lambda_C+\mu_C)}(1 - e^{-(\lambda_C+\mu_C)t})$$
$$= P_{ADn}(t)P_{BDn}(t)P_{CDn}(t) \quad (19)$$

The Markov model for the three components is not shown here but can be easily developed. Also note that we have:

$$P_{AUp,BUp,CUp}(t) = P_{AUp}(t)P_{BUp}(t)P_{CUp}(t)$$
$$P_{ADn,BUp,CUp}(t) = P_{ADn}(t)P_{BUp}(t)P_{CUp}(t)$$
$$\vdots$$
$$P_{ADn,BDn,CUp}(t) = P_{ADn}(t)P_{BDn}(t)P_{CUp}(t) \quad (20)$$

And the steady-state probability for all components down is $P_{ADn,BDn,CDn} = \frac{\lambda_A\lambda_B\lambda_C}{(\lambda_A+\mu_A)(\lambda_B+\mu_B)(\lambda_c+\mu_c)}$.

Again, since the total rate existing the state of all three components being down is the sum of repair rates of these three components, the corresponding frequency and duration time can be calculated as:

$$F(t) = F_{ADn,BDn,CDn}(t) = P_{ADn,BDn,CDn}(t)(\mu_A + \mu_B + \mu_c) \quad (21)$$
$$T = T_{ADn,BDn,CDn} = \frac{1}{\mu_A+\mu_B+\mu_C} \quad (22)$$

In PCA studies, the total probability of a set of contingencies that cause system problems need to be calculated. Usually, these contingencies may be up to the third order and can be treated as minimum cut sets for quantification.

Equations (15), (16), (19), and (20) indicate that ***the probability of a Markov state for the two- and three-component models in Figures 7 and 8 is simply the product of the probabilities of the individual components being at the corresponding states.*** This is not a coincidence and the rationale behind this is that the failures and repairs of these components (A, B, and/or C) are completely independent of each other. Therefore, the probability of both components being down is the product of the probabilities of individual components being down, and so on. This can, therefore, be extended to cases for quantifying the state probabilities for the system with arbitrary number of components and will significantly reduce the complexity of calculation of probability, frequency, and duration time.

3.3 Exact Quantification of System Problem Indexes

This section presents two methods for calculating the system problem indexes: one based on a fully-expanded Markov model while another one based on the collection of minimum cut sets. The associated difficulties are also discussed.

3.3.1 System Problem Indexes Based on a Full Markov Model

Theoretically, following the same train of thoughts above a Markov model can be built by considering all possible combinations of single outages, and a Markov state is represented by the statuses (e.g., Up and Dn for a bi-state component model but multi-state models can also be used) of all the grid components. An illustrative Markov model is shown in Figure 9, where the repairs are not included.

The system consists of M components and Component i has N_i failure modes. The tree-like Markov model starts from a system state without any component failure, i.e., the initial system state is $C_{(1,0)}C_{(2,0)}\cdots C_{(M,0)}$, where $C_{(i,0)}, i = 1,2,\cdots,M$, indicates the normal status of Component i, and the

transitions to other states that contain component failures are characterized in the figure. Each additional failure generates a new system state. A Markov state stops branching out and becomes an end state when either the state represents that all components are down or the state causes the system to fail. In other words, for the PCA studies, a state becomes an end state when it causes a system problem.

Note this model differentiates the order of failures. Therefore, the states with the same component failures but different orders can be combined in the PCA application[5]. Based on the fully expanded Markov model (repairs can be naturally included), the following procedures will lead to the exact quantification of the system problem probability, frequency, and duration.

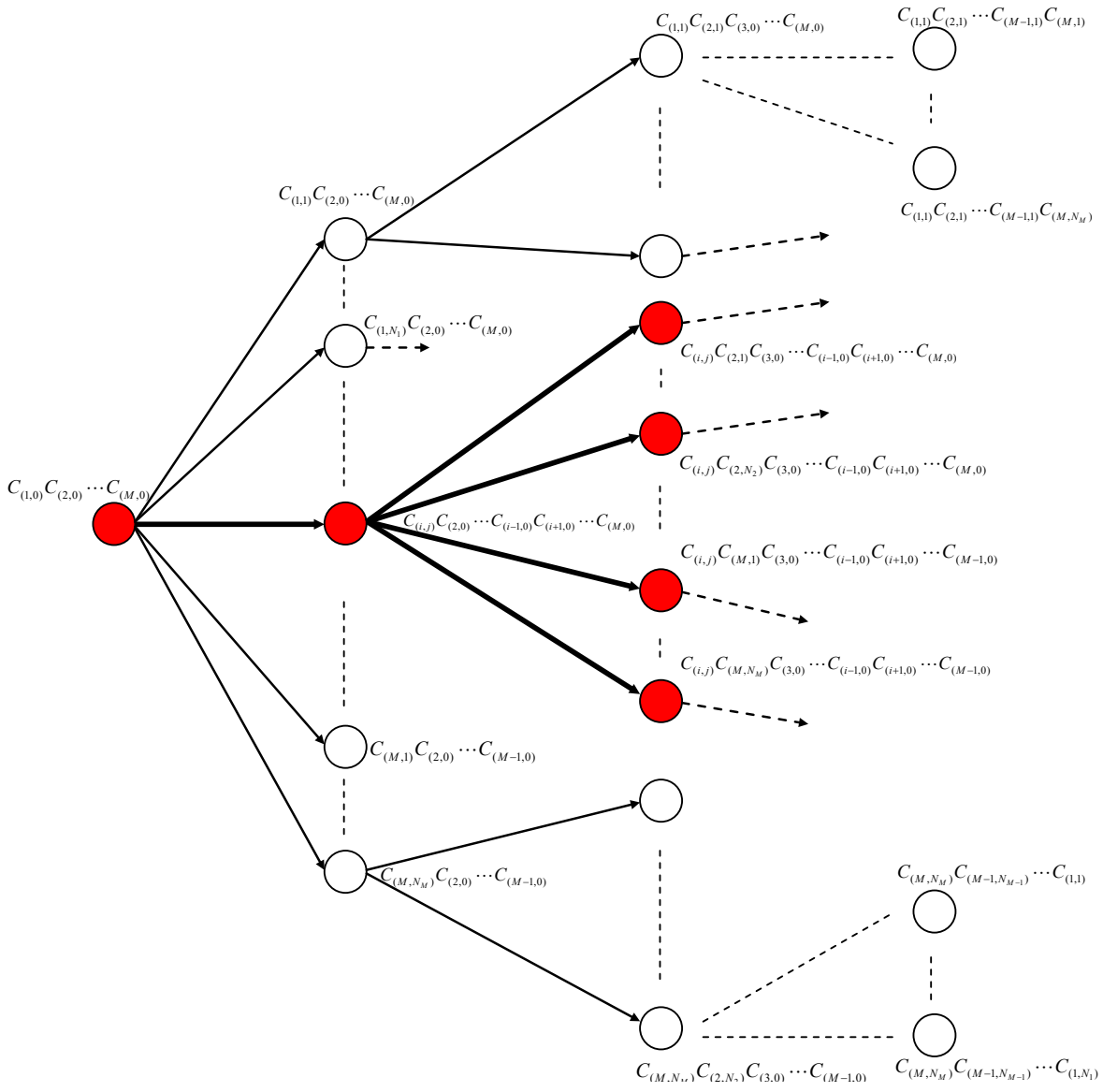

Figure 9: A full Markov model for a system consisting of M components (repairs not modeled).

[5] In PCA, for two contingency occurrences A and B, AB=BA, i.e., they occurred simultaneously.

Algorithm 1:

(1) Calculate the probability of each state causing a system problem by multiplying the probabilities of all components being at the status, similar to the probability calculation in Equations (16) and (20) for double and triple outages, e.g., $P_{C_{(1,0)}C_{(2,0)}\cdots C_{(M,0)}} = P_{1Up,2Up,\cdots,MUp} = P_{1Up}P_{2Up}\cdots P_{MUp} = (1-P_{1Dn})(1-P_{2Dn})\cdots(1-P_{MDn})$ based on Equation (12).

(2) The probability of a specific state causing the system problem and the transition rates exiting the state[6] are used to calculate the frequency of encountering this state using Equation (13).

(3) The probabilities and frequencies of states that cause system problems can be simply added together as the system problem probability P_{total} and frequency F_{total} since all these Markov states in the fully expanded model are exclusive.

(4) Finally, the duration time of the system problem can be directly calculated as:

$$T_{Duration} = \frac{P_{total}}{F_{total}} \tag{23}$$

(5) End

Note that major issue with this quantification is that Markov state probability calculation may not be as straightforward as in Equations (15) and (19). The calculation of a state probability needs the probabilities of all component status in the state (see the example in Step 1 above). This calculation is still manageable but is much exacerbated when CMOs need to be modeled.

In general, a CMO is modeled as a single contingency event. For example, the CMO of Components A and B can be a separate outage event with its own occurrence frequency and repair time. However, this CMO cannot be combined with the Component A outage or the Component B outage since they cannot occur together and are no longer independent, which changes the structure of the Markov model and caution is needed when formulating the Markov model. Therefore, the probability of the state is no longer simply the product of the probabilities of the individual components at the specific status, i.e., Equations (15), (16), (19), and (20) do not hold any more. The details are not discussed here but an example can be found in [Billinton 1981], where the steady-state probabilities of both components being down were calculated.

The only way to obtain the exact state probabilities is to solve the entire Markov model, which is generally intimidating and almost impossible for a system with a relatively large number of components. Without knowing the exact probabilities of the states, the exact quantification of system problem probabilistic indexes cannot be achieved.

3.3.2 System Problem Indexes Based on Contingency List

In general, quantifying the probabilities of each Markov state for a large system can be difficult since it needs to account for probabilities of all components that are up and all components that are down. It is more feasible to use the Cut set Probability Truncation method discussed in Section 2, e.g., consider only the single, double, and triple outage results. To calculate the system problem indexes caused by a collection of minimum cut sets (of the top event or the system problem) $MCS[Top]$, i.e., the contingencies

[6] Since this state causes the system problem, it becomes an end state. Therefore, the transition rates exiting from this state are only the repair rates.

of different orders, different procedures can be developed according to the methods that are being used to calculate the system problem probability.

If rare event or MCUB approximation methods are being used, then Algorithm 2 can be used:

Algorithm 2:

(1) Calculate the probabilities of outages of individual components, i.e., Component A, B, C, … using Equation (12) for single contingencies.
(2) Calculate the system problem probability P_{total} by accounting for the probabilities of the collection of the cut sets using either one of the approximation methods (rare event in Section 1 or minimum cut set upper bound approximation in Section 3).
(3) Calculate probabilities of individual cut sets $P_i, i \in MCS[Top]$ using Equations (12), (15), and (19) for single, double, and triple contingencies.
(4) Calculate the frequencies of individual cut sets $F_i, i \in MCS[Top]$ are calculated[7] using Equations (14), (17), and (21) for single, double, and triple contingencies based on result from Step (3).
(5) Calculate the system problem frequency by summing the frequencies of individual contingencies, i.e., $F_{total} = \sum F_i\ ,\ i \in MCS[Top]$.
(6) Calculate the system problem duration time by using Equation (23).
(7) End

Algorithm 2 based on rare event approximation is currently being used. Obviously, the calculated system problem indexes are not exact.

The exact quantification of system problem indexes can be done by using the BDD, as shown in Algorithm 3 below:

Algorithm 3:

(1) Calculate the probabilities of outages of individual components, i.e., Component A, B, C, … using Equation (12) for single contingencies.
(2) Create BDD diagram for the given list of cut sets and enumerate paths in the BDD diagram from the leaf nodes to the root node.
(3) For each of the paths,
    (3.1) Calculate the probability by multiplying the probabilities of the events along the path
    (3.2) Calculate the corresponding frequency of each path, i.e., the product of each path probability and sum of failure rates of all events that are not failed (the else branch) and the repair rates of all events variables that failed (the then branch) along the path[8].
(4) Calculate the total probability P_{total} and frequency F_{total} by summing the probability and frequency corresponding to each path
(5) Calculate the system problem duration time by using Equation (23).

[7] Again, the total rate exiting from a cut set is the sum of the repair rates of individual components involved in the cut set.
[8] This is based on Equation (13), similar to the step (2) in Algorithm 1. The states of individual events along a path can be considered a combined Markov state and the frequency can thus be calculated using Equation (13). Alternatively, the frequency of each path can be the product of each path probability and sum of failure rates of all events that are failed (the Then branch) and the repair rates of all events variables that not failed (the Else branch) along the path.

(6) End

Calculation of the total probability of top event is straightforward using BDD. The frequency calculation in Step (3.2) can be explained using the example BDD diagram for the top event in Figure 5:

$$P(Top) = P(x_1x_2) + P(x_1\bar{x}_2x_3) + P(\bar{x}_1x_2x_3)$$

The events involved in a path actually formulate a system state, i.e., the combination of the occurrences or not of the events. For path $\bar{x}_1x_2x_3$, $\bar{x}_1x_2x_3$ is also a state of the system. From this perspective, all the possible transitions to and from the state can be defined, as shown in Figure 10. These transitions are from and to different states. The transition rates from or to the state can be used to calculate the frequency to encounter this state, as shown in Equation (13). Note that in Figure 10, $\bar{x}_1$ means the success state of x_1 and x_2 and x_3 mean the failed state by following the convention in BDD representation.

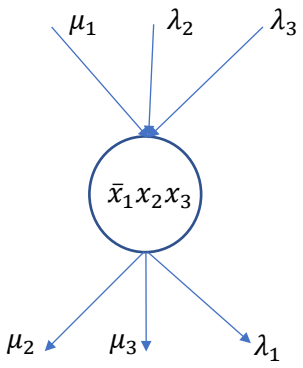

Figure 10: Transitions of an example state.

Taking the path that corresponds to this state $\bar{x}_1x_2x_3$ as an example, the top event will occur if x_1 does not fail, x_2 and x_3 both fail. The probability of this path $P(\bar{x}_1x_2x_3) = (1 - P(x_1))P(x_2)P(x_3)$ and the frequency is $P(\bar{x}_1x_2x_3)(\lambda_1 - \mu_2 - \mu_3)$ [Amari 2000], i.e., the frequency exiting this state (see Equation (13)).

Therefore, system problem indexes can be either approximately or exactly calculated by using different methods. If the BDD is used to quantify the probability of the system problem P_{total}, then the top event probability also becomes exact, just like the quantification method based on the Markov model. Similarly, the system problem frequency and duration time are also exact. A cut set can be considered an end state that is causing the system problem and the total probability of the collection of cut sets can be exactly calculated by adding the probabilities of the paths that are exclusive and lead to the root node in the BDD, i.e., the system problem, similar to the quantification of the probabilities of the Markov states. Therefore, the quantification methods based on the full Markov model and the cut sets can be equivalent and all exact.

Also note that Step (3) suggests that this quantification method also suffers a similar issue as using the full Markov model when CMOs are modeled. The exact probabilities of both (or three) components being down can still be calculated by solving the two- (or three-) component Markov model and three-component Markov model with CMOs modeled (if only up to triple contingencies are considered). The issue arises when the BDD is used since it will build the BDD diagram assuming that the CMOs are

independent events and include them in the BDD diagram and the paths leading to the root node. This can be potentially solved by removing the paths that consist of the CMOs and the single outages of the corresponding components in the CMOs. For example, if a path from a leaf node to the root node consists of a CMO of outages of Components A and B and the single outage of Components A and/or B, then the path is discarded since it is impossible. Therefore, it is still possible to calculate the exact probability. As long as the total probability is exact, then the calculated frequency and duration time are also exact.

4. Implementation of PCA Quantification Methods

Rare event approximation has been implemented. Its implementation is very straightforward and does not need any further discussion. MCUB, although not implemented in commercial PCA too, is also easy. The major focus here is on the BDD implementation. In this study, all these three methods have been implemented in Python scripts and a study is performed using an example list of contingencies to compare the conservativeness of the approximation methods and demonstrate the BDD capability of handling a large number of contingencies.

4.1 Existing Open-source BDD Software Packages

The key idea to implement a BDD data structure is to store an ITE node $ite(x, G, H)$ using a hash table consisting of a variable x and two addresses for G and H [Rauzy 1993]. Rather than develop our own BDD manipulation program, we sought to use existing reputable open-source packages.

Since the target application of this study is to quantify the system reliability indices beyond the probabilities of the contingencies, we were particularly interested in open-source BDD software packages that can be tailored for this purpose. Multiple open-source software libraries for BDD were identified, including Sylvan, Cudd and Tulip-dd. A comparison of these packages is shown in Table 1.

Table 1: Selection criteria for BDD software tool

Packages	Ease of usage, documentation, configuration	Quantification	Scalability	Additional functions	Track record
Sylvan	Very difficult to install because of the shared workers	No built-in quantification	High scalability (Parallelizable)	N/A	Between Cudd and Tulip
Cudd	C/C++ based, good documentation and relatively easy to setup, poor interface design and additional interface is needed to take input	No built-in quantification	High scalable by optimizing the BDD structure and memory	Deal with ZDDs (Zero-suppressed) DDs, built-in strategies to reduced BDD sizes via reordering	Well-established with many users
Tulip-dd	Python-based, very easy to install and use with flexible interface to parse input	No built-in quantification	Medium to high	Tulip is a Python library with C/Python (Cython) bindings to Sylvan and Cudd	The newest of the packages. Its main purpose is to

The major characteristics of each BDD package are also provided in Table 1. Scalability is key to applying to real problems with potentially thousands of variables found in power systems. However, as indicated above, the cut sets for reliability assessment in PCA are coherent, i.e., only AND and OR operations are involved and the BDD representation can be very efficient. In addition, the number of contingencies that need to be evaluated is more limited by the power flow computation effort. Therefore, the scalability should be manageable.

On the other hand, it appears that these tools were developed for circuit design applications and only create BDD diagrams. None of the tools has a built-in probability quantification function. Therefore, the

section below is dedicated to the development and implementation of the quantification schemes based on the BDD diagrams created by the existing tool(s).

4.2 A Recursive Algorithm for BDD Quantification

Theoretically, the BDD quantification is straightforward, i.e., it can be done by enumerating the individual paths from leaf nodes to the top event or root node and calculating the probability of each path. Since these paths are disjoint, the probabilities of individual paths can be summed as the total probability of the top event.

The main problem with enumerating BDD paths to compute the top event probability is that the number of paths grows exponentially with the number of variables. The existing BDD packages, although do not have built-in quantification functions, can generate the paths exhaustively and dump them into a text file. The text file can thus be loaded to calculate probabilities of individual paths and therefore, the total probability of the top event. However, text files with all the path elements of the BDD grew to terabyte order of magnitude once we got to more than 100 variables. Besides being very slow, the amount of memory required rendered this method extremely high, which renders the path enumeration an impractical solution. We thus sought a recursive algorithm that does not require storing paths.

A BDD is essentially a data structure that stores the precursor (or parent) and successor (or child) information. The top event probability can be calculated recursively. We denote $P(VARIABLE, t)$ the probability of node $variable$ (lower case) in the "**t**hen" branch, $P(VARIABLE, e)$ the probability of node $variable$ in the "**e**lse" branch. The top event probability is thus denoted as $P(VARIABLE)$. Then in the BDD, the probability of a $parent$ node can be calculated as:

$$P(PARENT,*) = P(parent)P(CHILD, t) + \big(1 - P(parent)\big)P(CHILD, e)$$

where "*" means either t or e. Note $P(parent)$ and $1 - P(parent)$ are the probabilities of occurrence and non-occurrence of variable $parent$. A child node may also be a parent node to its child nodes.

Following the above notation, the top event probability in Figure 5 is $P(X_1)$, $P(X_2, t)$ is the probability of node x_2 in the "then" branch of x_1, and $P(X_2, t)$ is the probability of node x_2 in the "else" branch of x_1, so that

$$P(Top) = P(X_1) = P(x_1)P(X_2, t) + \big(1 - P(x_1)\big)P(X_2, e).$$

$P(X_2, t)$ and $P(X_2, e)$ can be solved recursively until the leaf nodes are encountered, i.e.,

$$\begin{aligned} P(X_2, t) &= P(x_2)P(1, t) + (1 - P(x_2))P(X_3, e) \\ &= P(x_2) + (1 - P(x_2))P(x_3) \end{aligned}$$

and

$$P(X_2, e) = P(x_2)P(X_3, t) + \big(1 - P(x_2)\big)P(0, e)$$

$$= P(x_2)\big[P(x_3)P(1, t) + \big(1 - P(x_3)\big)P(0, e)\big]$$

$$= P(x_2)P(x_3)$$

Note that $P(0,e)$= $P(0,t)$=0 and $P(1,e)$= $P(0,t)$=1. Therefore, $P(Top) = P(X_1) = P(x_1)[P(x_2) + (1 - P(x_2))P(x_3)] + \big(1 - P(x_1)\big)P(x_2)P(x_3)$.

This implementation of such a recursive algorithm will be further described in section 4.3 below.

4.3 Scalable Implementation of Recursive BDD Quantification

The recursive algorithm discussed above, coupled with a *memoization* (or tabling) scheme to store the intermediary results in cache so they can be recalled rather than re-computed, enables a scalable implementation of the BDD quantification. Two such algorithms are described in [Rauzy 1993] and [Zang 2000]. We implemented the algorithm in [Zang 2000], shown below as pseudo-code:

```
Prob(F) {
        if (F == 0)
            return 1
        else if (F == 1)
            return 0
/* Memoization */
        else if (computed-table has entry {F, P_F})
            return P_F
/* recursive call F = ite(x, F1, F2), P(x) is given in the input */
        else {
             P_F = Prob(F1) + P(x) * (Prob(F2) – Prob(F1))
        }
        insert_computed_table({F, P_F})
        return P_F
}
```

Figure 11: Pseudo-code for the recursive quantification algorithm.

The algorithm was implemented in Python using the Tulip-dd BDD manipulation package. In the next section, we compare its performance to the approximation methods.

The Python implementation was validated by using the simple example shown in Figure. 5, which is easy to calculate the exact probability manually. The BDD drawn by the Tulip-dd is shown in Figure 12, which is essentially the same as the BDD in Figure 5 except that leaf nodes are combined into a single one here. With the hourly failure rates of 0.01, 0.02, and 0.03 and duration hours of 10, 20, and 30 for variables x_1, x_2, and x_3, the top event probabilities are calculated as 0.204, 0.194, and 0.180, respectively, using the rare event, MCUB, and BDD in the script.

Using Equation (12), we have $p_1 = 0.0909$, $p_2 = 0.2857$, and $p_3 = 0.4737$. Therefore, it is easy to verify that the top event probabilities calculated using Equation (3) for rare event approximation, Equation (5) for MCUB, and BDD for Figure 5 in Section 3 are exactly the same as the values calculated using the Python script above.

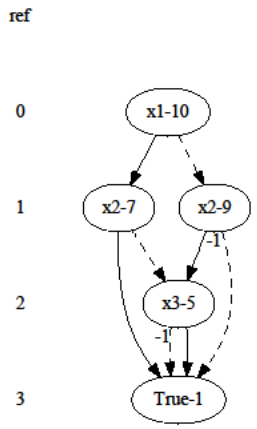

Figure 12: Reproduced BDD for $Top = x_1x_2 + x_2x_3 + x_1x_3$

4.4 Input Files

To create a standalone application that can be used to quantify the probability of a list of cut sets or contingencies that will cause certain system problems, we designed a simple I/O system with two input files. The first file is a csv file with the list of the contingencies. In this file, each separate contingency is represented in a single line by the single outages that it contains. Figure 13 shows an example of a cutset file snippet and its Boolean expression equivalent.

x1
x2,x3 $\leftrightarrow$ $y = x_1 + x_2x_3 + x_3x_4x_5$
x3,x4,x5

Figure 13: The cut sets file snippet on the left and its Boolean equivalent on the right.

The second file is also a csv file with a column for the single outage variables and two columns for their respective hourly failure rates and durations in hours. Two example input files are shown in Appendix A of this report.

The interface to handle the files intake was designed in Python so it could be directly handled in the Python/Tulip-dd environment. This simple I/O design can be adapted to interface with the APIs of a contingency analysis tool such as the PSS/E.

4.5 Comparing Exact and Approximate Methods

Rare event and MCUB (minimum cut set upper bound) approximation methods were also implemented to compare to the exact BDD solutions for system problem probabilities, frequencies, and durations, as discussed in Algorithms 2 and 3. Using a set of sample contingency cutsets, a comparison of calculated probabilities using these methods is presented in Table 2. The annual probabilities of system problems are calculated using the standalone application.

In the comparison study, cases considering a total number of 50, 150, 1,000 and 1,250 single outages were evaluated using the rare event, MCUB, and BDD methods, as shown in Table 2. The BDD diagram for the 50 contingency case is shown in Figure 14, where the event in the top is the top event that we are trying to quantify. The top event probability is the sum of each event probability along all paths from the leaf node 1 in the bottom of Figure 13 to the top event. The failure rates and durations of single outages were sampled from uniform probability distributions. We used a *Uniform(0, 1/500)* distribution for the hourly failure rate and a *Uniform(0, 50)* distribution for the duration in hours.

Table 2 shows exact solutions given by the BDD computation, while the approximations are more conservative, producing larger estimates. Frequencies and durations are also calculated and shown in Table 2. Note that frequencies are calculated using the Step 4 of Algorithm 2 for simplicity. Since the calculation times for rare event and MCUB approximates are negligible (less than half a second), they are not shown. Since the total probabilities of the cut sets or contingencies are determined not only by the probabilities of individual outages, but also whether different cut sets contain the same single outages, the numbers of single, double, and triple contingencies as well as how many of them contain the same single outages (overlap ratio) are also indicated in Table 2 for each case.

In Table 2, the probability of the rare event estimate is above 1 when 1,000 single outages are used. This nonsensical result obviously can be a big issue when the number of contingencies is large and/or the failure probabilities of individual outages are high. The MCUB, however, may offer a good compromise, as it is only 6% higher than the exact solution (.653 versus .610) while taking only tenths of a second as opposed to BDD's tens of seconds. On the other hand, the calculation times of BDD method for different cases appear manageable. In addition, the duration times calculated by using rare event approximation are also much conservative compared to MCUB and BDD methods. The most time consuming calculation is the contingency analysis. The time for BDD calculation can be acceptable.

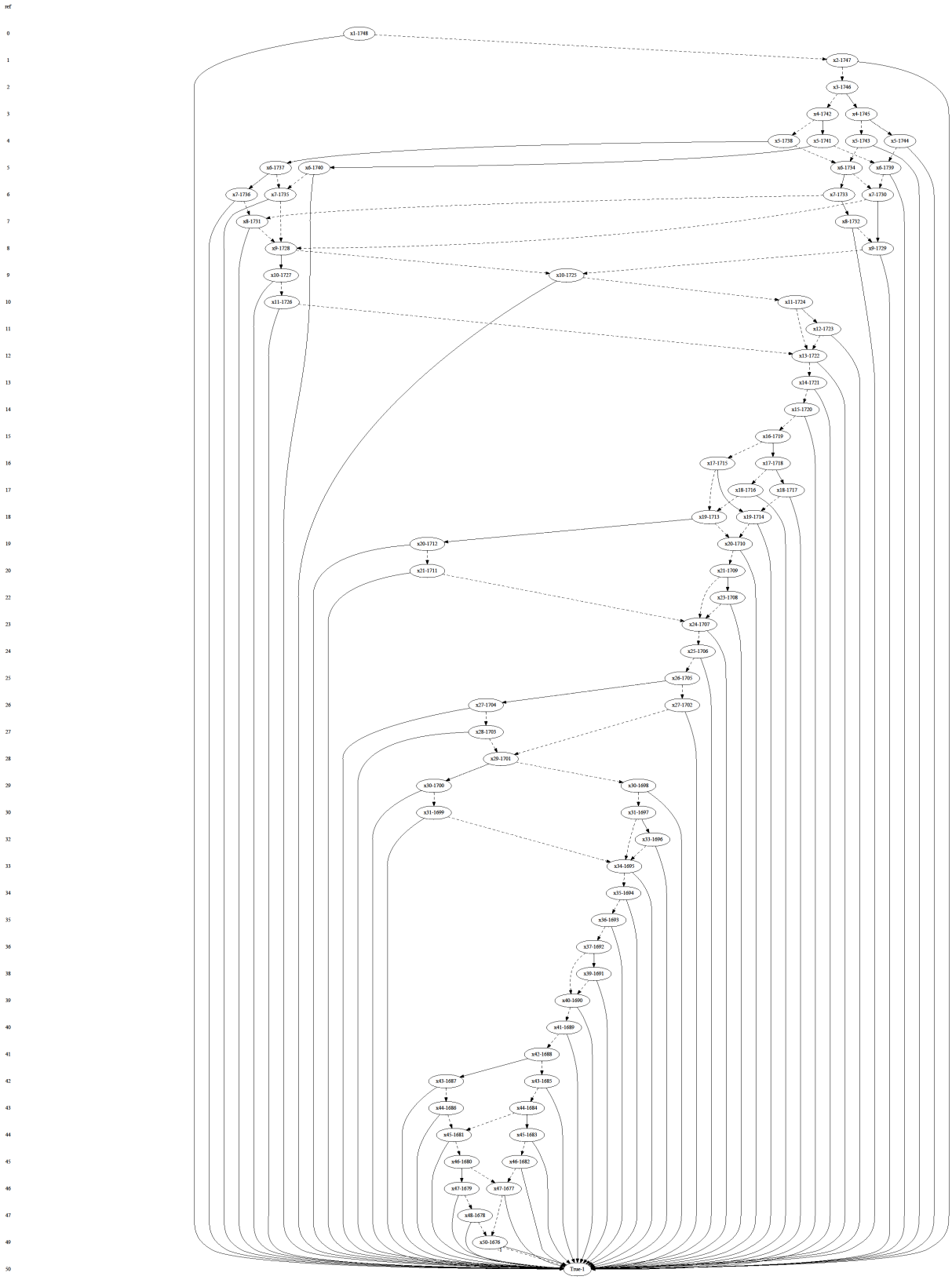

Figure 14: A BDD diagram for 50 contingencies

Table 2: Comparison of system problem index calculation

Number of single outages	Contingencies Structure; number of single/double/triple contingencies and overlap ratio	Computation time BDD (secs)	Probabilities			Frequency	Durations (hours)		
			Rare Event	MCUB	BDD	Total (per hour)	Rare Event	MCUB	BDD
50	20/30/0; overlap ratio ½	.056	.409	.341	.335	.0162	25.2	21.0	20.7
150	20/80/50; overlap ratio 2/3	0.70	.514	.407	.389	.0217	23.6	18.7	17.9
1,000	20/80/900; overlap ratio 2/3	28	.709	.514	.465	.0287	24.7	17.9	16.2
1,000	40/60/900; overlap ratio 2/3	29	1.046	.655	.610	.0420	24.9	15.6	14.6
1,250	80/120/1050; overlap ratio 2/3	44.4	1.384	0.756	0.710	.0560	23.2	12.7	11.9

In a summary, this comparison study shows that

(1) Both rare event and MCUB produce conservative results in terms of total probability and duration time while the BDD method can provide the exact solutions;
(2) The results using MCUB, although conservative, appear much less conservative than the results from the rare event approximation;
(3) The BDD method is more time consuming than the rare event and MCUB approximation but appear manageable and can be a feasible approach for the PCA quantification.

5. Summary

This report summarizes some of the existing top event quantification methods including the Sylvester-Poincare development and BDD that can be used to calculate the exact probabilities of Boolean functions, as well as the major approximation methods, such as the rare event approximation, truncation, and MCUB, which are frequently used in practice. Note that another method that can quantify the exact top event probabilities is the sum of disjoint products (SDP) by converting the cut sets into mutually exclusively disjoint products (see e.g., [Rauzy 2003]). The SDP method is not reviewed here because it suffers a similar combinatorial explosion issue as the Sylvester-Poincare development.

As indicated in the summary, the MCUB is a better approximation method than the commonly used rare event approximation, especially since MCUB does not lead to a probability of more than 1.0. Since the quantification of contingencies of different orders involves only AND and OR operations, the implementation of BDD in the quantification of contingency probabilities is feasible even for a large number of cut sets [Jung 2004]. Therefore, the MCUB approximation and BDD are better candidates for quantification applications in a PCA.

In addition to the probabilities of system problems, the theory and the procedures of calculating two other metrics including the frequencies and duration times are also discussed in detail in this report. It is shown that the system problem indexes can be possibly quantified with exact solutions using both the Markov model and cut sets, even with the presence of common mode failures.

Based on system reliability theory, two algorithms for quantifying the exact probabilities, frequencies, and durations of system problem indexes in PCA are developed. The MCBU and BDD methods have been implemented in this study by developing standalone Python scripts.

A demonstration of the proposed system problem indexes quantification was performed by using the rare event approximation, MCUB method, and the BDD method, respectively. For multiple cases with different number of contingencies, the results are consistent with what we expected, i.e., the rare event can produce very conservative outcomes, even with calculated probabilities larger than 1.0. On the other hand, the BDD method provides the exact result while it takes a longer time for calculation. The MCUB method may offer an acceptable trade-off, i.e., with conservative but reasonable results.

All these methods have been implemented in Python script that can be a standalone module to interface with a PCA tool and calculate system problem indexes using different methods. Since existing PCA tools use only the rare event approximation, the developed module provides useful and improved quantification schemes that enhance the state-of-the-art PCA analyses.

Future work may include the development of a user-friendly interface and post-processor to create load duration points from each case studied, then interpolate and/or extrapolate among these points to have an approximated load duration curve that can be overlaid by the calculated unserved energy using the same approach. In addition, a case study can be performed using the latest ePCA tool together with NERC guidelines and renewable generation data by considering both solar and wind generation in a real utility system.

References

[Amari 2000] S.V. Amari, "Generic Rules to Evaluate System-Failure Frequency," IEEE Transactions on Reliability, vol. 49, pp. 85-87, Mar. 2000.

[Billinton 1981] R. Billinton, T. K. P. Mledicherla, and M.S. Sachdev, "Application of Common-cause Outage Models in Composite System Reliability Evaluation," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-100, No. 7 July 1981.

[Bryant 1986] R. E. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, C-35-8, August 1986, pp. 677 – 691.

[Esary 1970] J. D. Esary and F. Proschan, "A Reliability Bound for Systems of Maintained and Independent Components," Journal of the American Statistical Association, 65 (1970), pp. 329 – 38.

[Jung 2004] W. S. Jung, S. H. Han, and J. Ha, "A Fast BDD Algorithm for Large Coherent Fault Trees Analysis," Reliability Engineering and System Safety, 83 (2004) pp. 369 – 374.

[Epstein 2005] S. Epstein and A. Rauzy, "Can We Trust PRA?" Reliability Engineering and System Safety, 88 (2005), pp. 195 – 205.

[Rauzy 1993] A. Rauzy, "New Algorithms for Fault Tree Analysis," Reliability Engineering and System Safety, 40 (1993) pp. 203 – 211.

[Rauzy 1997] A. Rauzy and Y. Dutuit, "Exact and Truncated Computations of Prime Implicants of Coherent and Non-coherent Fault Trees within Aralia," Reliability Engineering and System Safety 58 (1997) pp. 127 – 144.

[Rauzy 2003] A. Rauzy, E. Chatelet, Y. Dutuit, and C. Berenguer, "A Practical comparison of methods to assess sum-of-products," Reliability Engineering and System Safety, 79 (2003) pp. 33 – 42.

[Russell 1993] K. D. Russell and D. M. Rasmuson, "Fault Tree Reduction and Quantification – an Overview of IRRAS Algorithms," Reliability Engineering and System Safety, 40 (1993) pp. 149 – 164.

[Sinnamon 1996] R. M. Sinnamon and J. D. Andrews, "Fault Tree Analysis and Binary Decision Diagrams," Proceedings of IEEE Annual Reliability and Maintainability Symposium, 1996.

[Sperandio 2006] M. Sperandio and J. Coelho, "Constructing Markov Models for Reliability Assessment with Self-Organizing Maps," Proceedings of PMAPS 2006.

[Vesely 1981] W. E. Vesely, F. F. Goldberg, M. H. Boerts, and D. F. Haasl, "Fault Tree Handbook," NUREG-0492, U. S. Nuclear Regulatory Commission, January 1981.

[Way 2000] Y.-S. Way and D.-Y. Hsia, "A Simply Component-connection Method for Building Binary Decision Diagrams Encoding a Fault Tree," Reliability Engineering and System Safety, 70(2000) pp. 59 – 70.

[Yue 2019] M. Yue and J. Zhan, "Development of a Risk-informed Decision-making Capability Using Standard Electric Power Industry Planning Tools," A Technical Report Submitted to DOE AGM Program, July 31, 2019.

[Zang 2000] Zang, Xinyu, Hairong Sun, and Kishor S. Trivedi. "A BDD-based Algorithm for Reliability Graph Analysis." Department of Electrical Engineering, Duke University, Tech. Rep (2000).

Appendix A: Example Input Files for Quantification Demonstration

Several case studies were performed in Section 4. The two input files needed for one of the case studies are shown here for an illustration purpose. The input files for other case studies are similar.

An example input file of parameters for single outages is shown below. This .csv file can be read directly by the Python script. It contains variables representing the single outages and the corresponding the hourly failure rates (failr) and duration times (dur) in hours.

var	failr	dur
x1	0.000353	23.04639
x2	0.001439	11.44409
x3	0.000703	10.97336
x4	0.000211	28.62413
x5	0.001013	29.28446
x6	0.000572	15.69213
x7	0.001648	13.29514
x8	0.001249	0.687074
x9	6.43E-05	27.38387
x10	0.000614	2.479479
x11	0.000608	47.07507
x12	0.000307	47.60727
x13	0.001112	3.169638
x14	0.000451	37.6664
x15	0.000417	11.02617
x16	0.000868	25.54466
x17	0.001841	1.381362
x18	0.001429	41.87028
x19	0.001037	39.32516
x20	0.000421	46.06745
x21	0.001363	14.21914
x22	0.000232	41.26374
x23	0.000725	7.898564
x24	0.001176	49.75737
x25	0.000389	45.84436
x26	0.000537	40.21153
x27	0.001707	49.68438
x28	0.000376	38.39084
x29	2.04E-05	46.49559
x30	0.001476	35.59515
x31	0.000889	30.44849
x32	0.001277	30.11126
x33	0.000562	24.08951
x34	0.00032	44.92315
x35	0.000291	13.98515

x36	0.00093	9.264152
x37	0.001062	39.87017
x38	1.25E-05	37.75764
x39	0.001173	42.60311
x40	0.000401	35.51793
x41	0.000957	44.67263
x42	0.000335	0.359748
x43	0.000763	8.064783
x44	0.000628	49.42973
x45	0.000661	25.97955
x46	0.001556	11.25907
x47	0.000764	12.90673
x48	0.000311	34.38296
x49	0.001634	20.08437
x50	0.000792	10.25143

This example input file contains a list single and double contingencies consisting of the single outages above. The .csv file will be read by the Python script to create the corresponding BDD for the union of these contingencies and perform quantification of the total probability, frequencies, and duration times using BDD, rare event approximation, and MCUB method.

x1	
x3	x5
x2	
x4	x6
x5	x7
x6	x8
x7	x9
x6	x8
x10	
x7	x9
x11	x12
x9	x11
x12	x13
x14	
x13	x15
x14	x16
x13	
x15	
x14	x16
x17	x19
x16	x18
x20	
x19	x21

x20 x22
x21 x23
x20 x22
x24
x21 x23
x22 x24
x23 x25
x24 x26
x25
x27
x26 x28
x30
x27 x29
x28 x30
x29 x31
x30 x32
x31 x33
x35
x32 x34
x33 x35
x34
x36
x35 x37
x36 x38
x40
x37 x39
x38 x40
x39 x41
x40 x42
x41
x43
x42 x44
x43 x45
x44 x46
x45
x47
x46 x48
x50
x47 x49
x48 x50