A Hybrid Approach for a Multi-unit Probabilistic Risk Model using the Direct Quantification of a Fault Tree using Monte Carlo Simulation and the minimal cutset method

Shota Soga[a*], Eishiro Higo[a], Hiromichi Miura[a]
[a]Central Research Institute of Electric Power Industry, Kanagawa, Japan

Abstract: The importance of multi-unit probabilistic risk assessment (MUPRA) has been globally acknowledged since the Fukushima Daiichi Nuclear Power Plant accident. However, many challenges exist in implementing a practical MUPRA. One of the challenges is that MUPRA suffers exponentially increasing computational costs due to the increase in the number of multi-unit sequences and their heading events as the number of units increases. In this paper, we propose an approach that hybridize the direct quantification of a fault tree using the Monte Carlo simulation (DQFM) method and the minimal cutset (MCS) method. The proposed approach utilizes the DQFM method to quantify an MUPRA model and to identify dominant accident sequences. Then, the propose approach applies MCS method for identified dominant accident sequences. This hybrid approach overcomes a disadvantage of the DQFM method incapable of enumerating dominant MCSs. Also, we introduce an algorithm for the DQFM method to estimate the required importance measures, a key step in reducing the computational cost and making MUPRA more practical.
The proposed approach has two steps in quantification. The first step is quantifying an MUPRA model using the DQFM method. This allows us to obtain the multi-unit core damage frequencies (CDFs), the required importance measures, individual accident sequence frequencies, and their Monte Carlo errors. Estimated accident sequence frequencies provide which multi-unit sequences are dominant. In the second step, the MCS method is applied only to the estimated dominant multi-unit sequences to obtain the dominant MCSs.
The computational cost to estimate CDF by the DQFM method is roughly proportional to the number of units. Thus, the computational cost for an MUPRA model using the DQFM method is inexpensive. The DQFM method's ability to identify the dominant multi-unit sequences is a significant advantage, as the proposed approach allows us to generate MCSs using the MCS method for only these dominant sequences, rather than for all multi-unit sequences. This feature of the proposed approach can significantly reduce the overall computational cost of quantifying an MUPRA model.

Keywords: Direct Quantification of Fault Tree using Monte Carlo Simulation, Multi-Unit Probabilistic Risk Assessment, Minimal Cutset method.

1. INTRODUCTION

Probabilistic risk assessment (PRA) of nuclear power plants (NPPs) is important in improving NPP safety and economic efficiency. PRA has been applied to various risk-informed applications such as risk-informed technical specifications initiatives [1]. Applications of PRA are not limited to a single unit. It can assess the risk of multiple units and a site. The Fukushima Daiichi NPP accident showed the importance of multi-unit PRA (MUPRA), and PRA researchers and engineers have been working on establishing a practical MUPRA approach. Internationally, extensive research has been conducted to develop MUPRA such as [2].

To assess the probabilistic risk of an NPP, PRA engineers build a logic model consisting of fault trees and event trees. Then, this logic model is solved by the minimal cutset (MCS) method, binary decision diagram, or their variants. Both methods have advantages and disadvantages, but a modern PRA model accounts for complex systems and dependencies that result in high computational costs. Typically, an MUPRA model is built based on a combination of single-unit PRA models, and thus, an MUPRA model has higher computational costs than single-unit PRA models.

Besides these two primary methods, there is also a method that statistically solves a PRA model by Monte Carlo simulation. This method is called "direct quantification of fault tree using Monte Carlo simulation (DQFM)." Originally, the DQFM method was proposed and developed by the Japan Atomic Energy Research Institute [3]. Its name only refers to a fault tree but can also be applied to an event tree model. The DQFM method's main advantage is its ability to account for success events in a fault tree and an event tree with a small

computational cost. On the other hand, the MCS method only considers combinations of failure events and excludes success events. A binary decision diagram can consider the success events but suffers exponentially increasing computational costs with the number of basic events. One of the other advantages of the DQFM method is its ease of parallelization. Thus, the DQFM method can utilize a modern multi-core CPU. In addition, the DQFM method has an advantage in interfacing with level 1PRA and level 2 PRA because the states of all basic events are determined in each Monte Carlo trial. Therefore, it is free from concerns about the computational costs of linking level 1 and level 2 PRA. It is also worth mentioning that the DQFM method can directly estimate a mean core damage frequency (CDF) by sampling basic event probabilities in each simulation without recovering an uncertainty distribution of CDF. The DQFM method can calculate a point estimate of CDF by fixed basic event probabilities in simulation.

However, the DQFM method has disadvantages as well as advantages. One of its disadvantages is that the DQFM method is not good at enumerating MCSs. In each Monte Carlo trial of the DQFM method, the method generates a set of basic events that are failure. When this set makes a fault tree's top event false, this set is cutset but not always MCS. Modern PRA applications are based on MCSs; therefore, it is difficult for PRA engineers to shift the DQFM method from the MCS method. Also, it is computationally expensive to estimate importance measures, especially when the number of basic events is large. It is a clear disadvantage that the DQFM method is subject to a statistical error even if this statistical error is relatively small compared with CDF and reducible in most cases. Furthermore, it often fails to quantify very rare accident sequences. In addition, the DQFM method is not suitable for recovering uncertainty distribution.

This paper proposes a hybrid approach that utilizes the DQFM method and the MCS method. The proposed approach has two steps. The first step is that the DQFM method statistically solves an MUPRA model and identifies the dominant accident sequences for MCS analysis. This step lets us obtain the multi-unit CDF, multi-unit sequence frequencies, the required importance measures, and their Monte Carlo errors. If one only needs those values, one can finish estimation in this step. If one needs MCSs for PRA applications, the second step performs an MCS analysis on obtained dominant multi-unit sequences. The generated MCSs allows us to use existing PRA platform.

Section 2 describes the basic algorithm of the DQFM method and its extension to an event tree. In addition, we describe the algorithm that uses the DQFM result as a truncation value for MCS analysis. Section 3 describes the application of the DQFM method to MUPRA. Section 4 concludes this study and provides future research.

2. The Hybrid Approach of the DQFM Method and the MCS Method

This chapter briefly describes the basic algorithm of the DQFM method and how to generate the MCSs utilizing the DQFM result. Then, we propose a hybrid approach to quantifying an MUPRA model. We also propose a simple bottom-up algorithm to estimate importance measures.

2.1. Basic Algorithm of the DQFM method

The DQFM method estimates CDF using Monte Carlo simulation. To understand its concept, let us assume a simple fault tree with one basic event having a failure probability equal to 0.05, as shown in Figure 1.

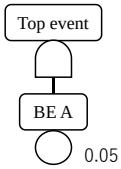

Figure 1. A simple fault tree with one basic event

The DQFM method determines Boolean states of basic events using the Monte Carlo simulation to estimate a top event frequency. In Monte Carlo simulation, a random value from a uniform distribution of interval $(0,1)$ is generated for each basic event in each Monte Carlo trial and compared with a basic event probability. If a random value is less than the basic event probability, then the Boolean state of that basic event is set to true.

If a random value is greater than or equal to the basic event probability, then the Boolean state of that basic event is set to false. Figure 2 visualizes this Monte Carlo simulation.

A fault tree becomes a Boolean expression and the top event's Boolean state can also be determined once all basic events' Boolean states are determined. Now, an estimated top event frequency P_{est} is given by the number of true top events N_{true} divided by the total number of Monte Carlo simulations N_{Monte}. Algorithm 1 summarizes the DQFM method. Algorithm 1 counts the number of times that a top event is true, N_{true}, and Eq. (1) and (2) provide an occurrence frequency of a fault tree and its Monte Carlo error.

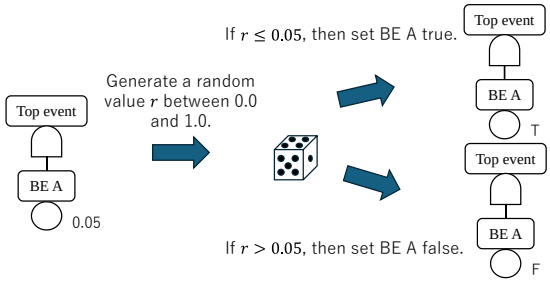

Figure 2. Process to determine a Boolean state of a basic event using Monte Carlo simulation

$$P_{\text{est}} = \frac{N_{\text{true}}}{N_{\text{Monte}}} \tag{1}$$

In addition, a Monte Carlo error σ_p of P_{est} is obtained by

$$\sigma_p = \sqrt{\frac{p_{\text{est}}(1 - p_{\text{est}})}{N_{\text{Monte}}}} \tag{2}$$

Algorithm 1: DQFM method to quantify a fault tree

Inputs:
$\boldsymbol{s}$: The set of basic events in a fault tree
$P(s_i)$: The probability of the ith basic event
$F(\boldsymbol{s})$: A fault tree for evaluation
N_{Monte}: Number of Monte Carlo trials

Outputs:
N_{True}: Number of counts that the top event is true.

Initialization
$N_{\text{True}} = 0$
for $i = 1$ **to** N_{Monte} **do**:
  for $k = 1$ **to** length($\boldsymbol{s}$) **do**:
    Generate a random number r in (0,1)
    If $r \leq P(s_i)$ **then**
      Set the state of s_i true.
    else
      Set the state of s_i false.
    end
  end
  Evaluate $F(\boldsymbol{s})$
  If $F(\boldsymbol{s})$ is true, **then**
    $N_{\text{True}} = N_{\text{True}} + 1$
  end
end
Return N_{FT};

Algorithm 1: Pseudo algorithm of the DQFM method to evaluate a fault tree

Algorithm 1 fails if the number of Monte Carlo trials is insufficient. This can happen when a top event is rare, or the number of Monte Carlo trials is inadequate. One can resolve the latter case by increasing the number of Monte Carlo trials. However, increasing the number of Monte Carlo trials to capture a low-occurrence event may not be possible for the former case. Therefore, one must apply variance reduction techniques such as importance sampling techniques in such a case. However, if these techniques are used, the algorithms described in this paper are no longer applicable. In addition, variance reduction techniques cannot evaluate a whole event tree like the DQFM method. If one applies variance reduction techniques, then all multi-unit accident sequences are individually quantified. Thus, variance reduction techniques are useful to estimate selected low frequent multi-unit accident sequences but not suitable for estimating overall CDF.

One can easily extend Algorithm 1 to an event tree. First, all the heading events in an event tree are evaluated using the sampled Boolean states of the basic events. Next, given the Boolean states, the end state of an event tree is determined in each simulation, as shown in Figure 3. Algorithm 2 summarizes the modified algorithm. Note that Algorithm 2 returns the occurrence frequencies of all the end states of an event tree.

In addition, Algorithm 2 is also designed to return each end state frequency neglecting success branches. Thus, Algorithm 2 can estimate both true end state frequencies ($N_{\text{true},k}/N_{Monte}$) and end state frequencies neglecting success branches (($N'_{\text{true},k}/N_{Monte}$). The latter values are good estimates of accident sequence frequencies based on MCS analysis because MCSs do not consider success branches whereas the former value is good estimates of true end state frequencies.

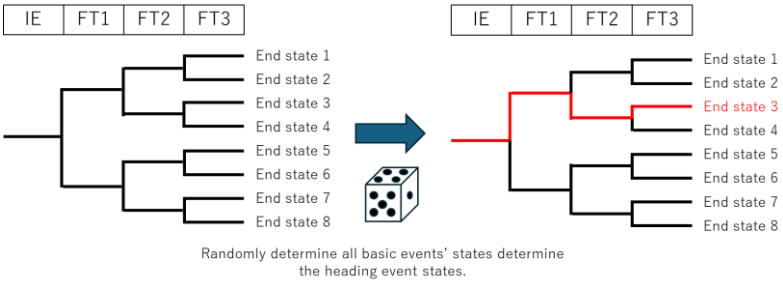

Figure 3. Determination of the end state of an event tree using Monte Carlo simulation

2.2. Identification of Multi-unit Dominant Sequences and MCS Generation

Algorithm 2 can evaluate the occurrence frequencies of all the end states of an event tree. One can utilize this information to rank the end-states of an event tree and for the truncation value of MCS analysis. Note that estimated end-state frequency $P_{\text{est},k} = N_{\text{true},k}/N_{\text{Monte}}$ accounts for success branches. However, MCS analysis typically does not consider success branches, and therefore, this estimated end-state frequency may be different from the frequency estimated by the MCS method. If success branches are neglected, it is possible that MCSs include logically impossible combinations such as a failure of a component that succeeds in a success branch. The validity of these MCSs can be checked by making basic events in a MCS true and the other basic events are false whether the same accident sequence is traced. If not, such a MCS is logically impossible and removed.

For the above reason, the estimated end-state frequency is not a good candidate for a truncation value for the MCS method. Hence, one needs an end-state frequency without success branch consideration for a truncation value of the MCS method. For this purpose, Algorithm 2 also provides $N'_{\text{true},k}$ that remove the effect of success branches. It is important to note that a true end-state frequency is not always appropriate for a truncation value of MCS analysis. To the authors' best knowledge, there is no approach to predict the number of MCSs given a truncation value. Algorithm 3 shows an automated iterative try-and-error approach to obtain the user defined number of MCSs [4]. This automated algorithm helps reduce the cost of PRA engineers performing try-and-errors to find appropriate truncation values for all end states.

Algorithm 3 begins with an initial guess of a kth end-state frequency with a 99.86% probability that a kth end-state frequency without success branch consideration is less than or equal to this initial guess. This initial guess is a good initial guess for MCS analysis. For example, let us assume an MCS has the largest value. The DQFM

method repeatedly simulates the states of all basic events and sets the top gate true if all basic events in this MCS are true. Hence, the initial guess obtained by the DQFM method has a 99.73% probability greater than the end-state frequency guaranteed to be greater than or equal to the largest MCS.

Algorithm 2: DQFM method to quantify an event tree

Inputs:
$\boldsymbol{s}$: The set of basic events in an event tree
$P(s_i)$: The probability/frequency of ith basic event
$F_j(\boldsymbol{s})$: The fault tree of jth heading event
$E(\boldsymbol{s})$: The Event tree for evaluation.
N_{Monte}: The number of Monte Carlo trials
N_{he}: The number of heading events
N_{es}: The number of end states

Outputs:
$N_{\text{true},k}$: The number of counts that the k th end state is true
$N'_{\text{true},k}$: The number of counts that the k th end state without success branch consideration is true

```
Initialization
N_true,k = 0 and N'_true,k = 0 for k = 1 to N_es
Process
for i = 1 to N_Monte do:
    for k = 1 to length(s) do:
        Generate a random number r in (0,1)
        If r ≤ P(s_i) then
            Set the state of s_i true.
        Else
            Set the state of s_i false.
        End
    end
    for j = 1 to N_he do:
        Evaluate F_j(s) to determine its top event state.
    end
    Determine the index of an end state k that is true by evaluating E(s) using evaluated F_j(s)
    N_true,k = N_true,k + 1
    for k = 1 to N_es do:
        If all heading events in failure branches of the kth end state are true, then
            N'_true,k = N'_true,k + 1
        end
    end
end
Return N_true,k and N'_true,k
```

Algorithm 2: Pseudo algorithm of the DQFM method to evaluate an event tree

Algorithm 3 iteratively adjusts a truncation value until a user-defined number of MCSs is generated. Note that PRA engineers may not need MCSs that are too small, such as MCSs less than 10^{-10} unless they need to evaluate importance measures. In such a case, one can terminate this iterative step without generating MCSs.

2.3. Importance Measure Estimation

The importance measure is typically estimated using generated MCSs. This estimated importance measure is sometimes inaccurate for basic events with very low probability because MCSs with such basic events are truncated during MCS analysis. In addition, an approximation such as a rare-event approximation also results in inaccurate estimation. The issue is that one cannot know how inaccurate the estimated importance measure by MCS analysis is. The DQFM method can resolve this issue. The DQFM method can evaluate an importance measure for a target basic event using Algorithm 1 with a small modification. However, it is computationally expensive if the DQFM method is applied to obtain importance measures for all basic events because it requires reevaluating a whole Monte Carlo simulation for all basic events.

Thus, the DQFM method is not good at estimating all the importance measures. Instead, we propose a bottom-up approach to evaluate an importance measure to ease this issue. Let us assume that a simple fault tree is shown in Figure 4, and we evaluate a risk achievement worth of a basic event BE B. Note that a risk achievement worth of a basic event $\boldsymbol{x}$ for a fault tree is defined as

$$\mathrm{RAW} = \frac{P(\mathrm{FT}|x = 1)}{P(\mathrm{FT})}$$

where $P(\mathrm{FT})$ is the probability or frequency of a fault tree and $P(\mathrm{FT}|x = 1)$ is the conditional probability of a fault tree given a basic event $\boldsymbol{x}$ fails. In the first step of the bottom-up approach, we preprocess a fault tree to obtain all branches from a target basic event to a top event (step 1), as shown in Figure 4 (left). Figure 4 (left) shows a case where the target basic event is BE B. In Figure 4 (left), there are two green branches toward the top gate. These two branches must be reevaluated for importance measure estimation.

We use these branch data to reduce the computational cost of reevaluating a fault tree by a simulation. In the next step (Step 2), a Monte Carlo trial determines all basic event states and evaluates the top event state, as shown in Figure 4 (right). In this example, alphabets T and F indicate the true and false states of basic events and gates.

Algorithm 3: MCS generation for the *k*th end state	
Inputs:	
$P'_{\mathrm{est},k}$: The estimated frequency of the kth end state without success branch consideration obtained from Algorithm 2	
$F_k(s)$: The fault tree with an AND gate that connects fault trees for the failure branches in the branch leading a kth end state	
N_{MCS}: Target number of MCSs for each end state	
Outputs:	
MCS_k: Generated MCSs for the kth end state	
1	*Initialization*
2	Set an initial truncation value $P_{\mathrm{trun}} = P'_{\mathrm{est},k} + 3\left(P'_{\mathrm{est},k}\right)^{0.5}$ and $N_k = 0$
3	*Process*
4	**while** $N_k \leq N_{\mathrm{MCS}}$ **do**:
5	Try to generate MCS_k for $F_k(s)$ with the truncation value P_{trun}
6	**If** MCS analysis timeouts, **then**:
7	$P_{\mathrm{trun}} = P_{\mathrm{trun}} * 1.01$
8	**else**
9	$N_k = \mathrm{count}(\mathrm{MCS}_k)$ and $P_{\mathrm{trun}} = 0.95 * P_{\mathrm{trun}}$
10	**end**
11	**end**
12	**return** MCS_k

Algorithm 3: MCS generation utilizing the DQFM result

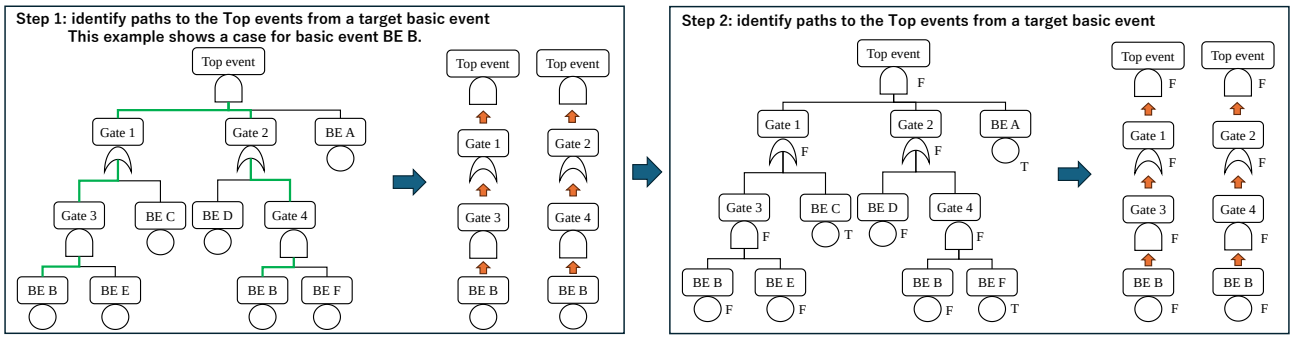

Figure 4. Example of the identification of branches to the top gate and evaluation of the fault tree

Then, there are several conditions when a fault tree is reevaluated. If the state of BE B is true, no reevaluation is required for the risk achievement worth. However, if the state of BE B is false, reevaluation is needed regardless of the top event state. This is because a fault tree may contain a logical negation operator, such as a NAND gate. Therefore, given the state change of a target basic event, the top gate's state may change from true to false. For each identified branch, we change the state of the bottom target basic event to true and evaluate the gates in the branch (Step 3), as shown in Figure 5.

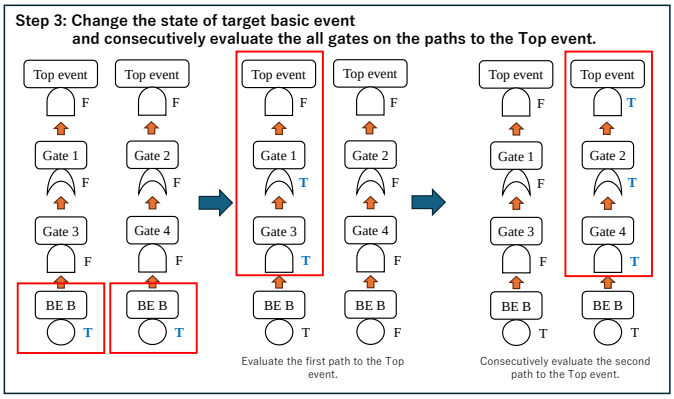

Figure 5. Example of reevaluating identified branches to the top gate

The proposed bottom-up approach becomes effective, especially when the size of a fault tree is large, and the size of identified branches is small because the cost of the reevaluation is small for the identified branches. However, for a small fault tree, the proposed approach does not improve the performance of the importance measure calculations.

3. APPLICATION TO MUPRA

The computational cost of the DQFM method linearly increases as the number of gates and basic events increases. This characteristic is desirable for MUPRA because the number of gates and basic events also almost linearly increases as the number of units increases. In addition, the Monte Carlo simulation is easily parallelized and, therefore, can utilize many cores to mitigate the increase in the computational cost. For example, one can use cloud resources or recent many-core CPU architecture for the DQFM method. In contrast, improving the efficiency of algorithms such as MCS calculations and binary decision diagrams is not easy.

As noted in Section 2.2, the DQFM method can identify dominant sequences of an event tree. Likewise, evaluating individual units in each Monte Carlo trial can identify the dominant combinations of multi-unit accident sequences. Evaluating the only dominant multi-unit sequences by the MCS method can reduce the number of multi-unit accident sequences for MCS analysis, reducing the overall costs. Note that there is a change that the identification of the dominant sequences fails because the number of Monte Carlo trials is insufficient. To avoid this issue, one needs to carefully consider the required number of trials and review the Monte Carlo error of each multi-unit sequence frequency.

To extend the DQFM method to MUPRA, one must consider inter-unit interactions summarized in reference [5-8] and modify single-unit models accordingly. Understanding how to evaluate these inter-unit interactions in an MUPRA model is the most important aspect of MUPRA. Algorithm 4 shows a pseudo algorithm describing the DQFM method applied to MUPRA for frequency estimation. Algorithm 4 first determines an accident sequence of each unit by Monte Carlo trial and combines them as a multi-unit accident sequence. Then, the algorithm repeats this trial to count multi-unit sequences. This simple algorithm can be extended to evaluate the importance measures described in Section 2.3 and to generate MCSs for dominant sequences.

3.1. Numerical example of Algorithm 4

This section demonstrates the DQFM method applied to a large event tree model with a structure like an MUPRA model. Since no benchmark MUPRA model is publicly available, we built a simple base event tree with five functions and three redundant systems per function for a unit, as shown in Figure 6. Note that each fault tree does not represent any actual safety system but is randomly generated. Each function must succeed to achieve a safe shutdown. Then, we copied the whole structure of this base event tree and associated fault

trees, modified the fault trees, and assigned different basic event names and probabilities. Also, we injected common basic events among these event trees, which work as inter-unit interactions between and among event trees.

Algorithm 4: DQFM method applied to MUPRA

Inputs:

N_{unit}: Total number of units
$\boldsymbol{s}$: Set of all basic events considered in MUPRA. This includes the result of multi-unit initiating event analysis
$E_i(\boldsymbol{s})$: Event tree model of ith unit
N_{Monte}: Number of Monte Carlo trials
N_{MCS}: Target number of MCSs for each end state

Outputs:

$\text{P}(\boldsymbol{c})$: Map from the combination of accident sequences $\boldsymbol{c}$ to its occurrence frequency

Initialization
Set $\text{P}(\boldsymbol{c})$ empty map and **MCS** = { }
Process
for $j = 1$ **to** N_{Monte} **do**:
    Determine the states of $\boldsymbol{s}$ using Monte Carlo simulation.
    for $i = 1$ **to**N_{unit} **do**:
        Evaluate $E_i(\boldsymbol{s})$ to determine an ith unit's end state c_i
    end
    If inter-unit interactions are not modeled in an MUPRA model and any accident sequence results in a core damage state, **then**
        Apply inter-unit interactions and reevaluate accident sequences c_i
    end
    Set $\boldsymbol{c} = \{c_1, c_2, \cdots, c_{N_{\text{unit}}}\}$
    If $\boldsymbol{c}$ is in $\text{P}(\boldsymbol{c})$**, then**
        Increment $\text{P}(\boldsymbol{c})$ by one
    Else
        $\text{P}(\boldsymbol{c}) = 1$
    end
end
Set $\text{P}(\boldsymbol{c}) = \text{P}(\boldsymbol{c})/N_{\text{Monte}}$ for all keys in $\text{P}(\boldsymbol{c})$
Return $\text{P}(\boldsymbol{c})$

Algorithm 4: the DQFM method applied to an MUPRA model

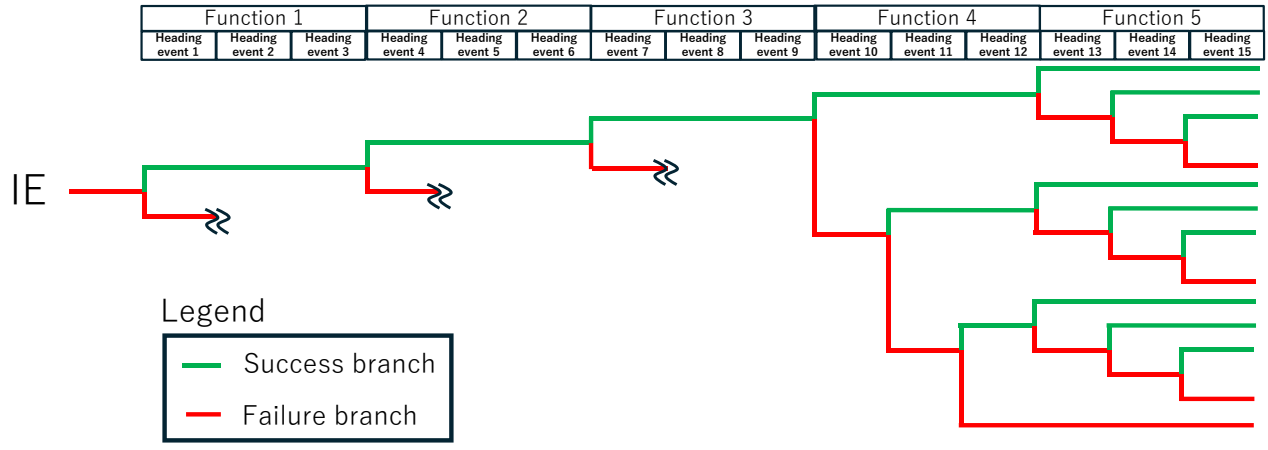

Figure 6: A base event tree structure

Table 1. Characteristics of the base event tree model with 10^7 Monte Carlo trials for the DQFM method

	Base Event tree
# of BEs	1227
CDF (DQFM)	2.4955E-03
MCSE(DQFM)	1.5777E-05

Figure 7 shows the elapsed time given the number of the Monte Carlo trials for the base event tree. It is clear that the elapsed time almost proportionally increases as the number of the Monte Carlo trials increases. Figure 8 shows the estimated CDF and its error bounds of $3\sigma_{MCSE}$. As the number of the Monte Carlo trials increase, the error bound decreases.

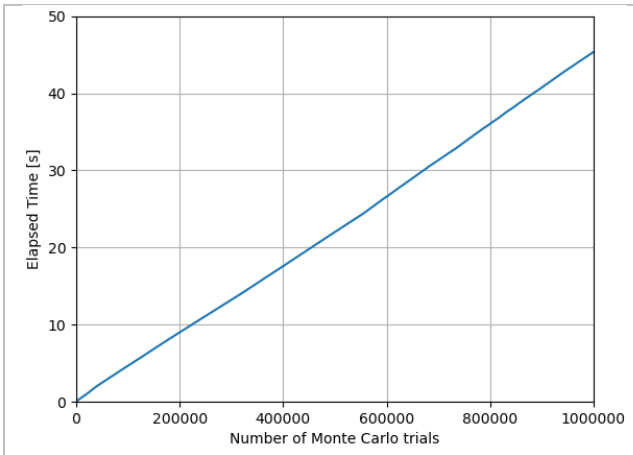

Figure 7. Elapse time of the DQFM method for the base event tree

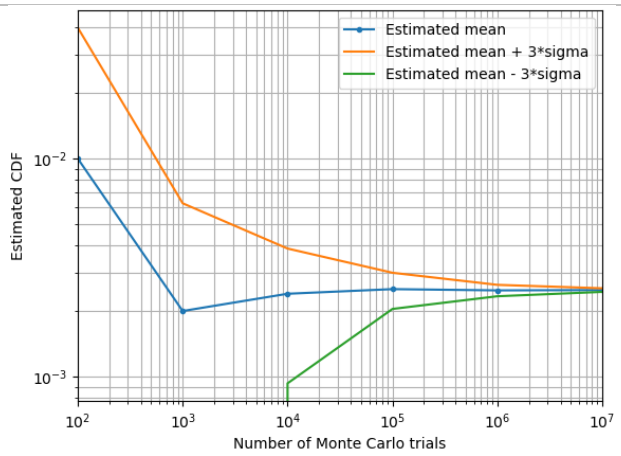

Figure 8. Estimated CDF with its error band for the base event tree

Figure 9 shows the elapsed time required to evaluate the hypothetical MUPRA model generated above. The hypothetical MUPRA model has a lower CDF value compared with a single-unit one. Therefore, the MUPRA model requires many Monte Carlo trials as shown Figure 9. This is a strong drawback of the DQFM method applied to a MUPRA model. To overcome this difficulty, one need to use multicore CPUs or general-purpose graphics processing unit programming. Figure 10 shows the required time to evaluate the MUPRA model with 10^7 Monte Carlo trials. Figure 10 shows the linearly increasing elapsed time as the number of units increases, and this property is desirable for a site with many units.

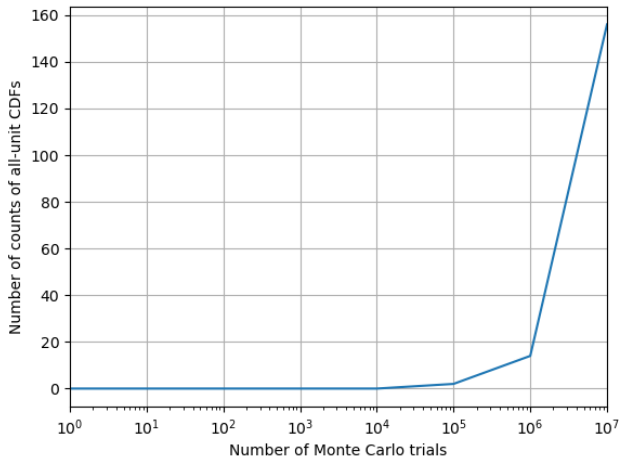

Figure 9. Number of counts that all four event trees are in core damage states

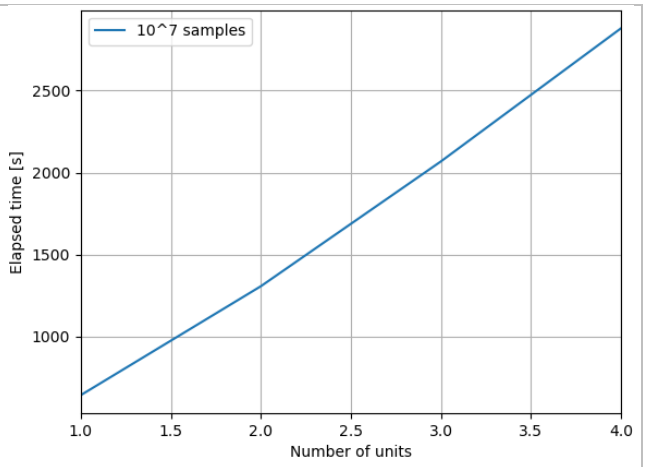

Figure 10. Elapsed time of the DQFM method given the number of units

3.2. Limitation of the DQFM method to MUPRA

The DQFM method for MUPRA easily fails when a combination of accident sequences has a very low frequency, such as 10^{-30}. In this case, the DQFM method fails to generate a good candidate of a truncation value and thus fails to generate an MCS. As mentioned in Section 2.1, variance reduction techniques are research direction to solve this issue.

The importance of such extremely rare accident sequences is low in risk-informed applications unless there are a significant number of rare accident sequences, and its aggregation contribution is not negligible. Thus, the DQFM method fails to quantify each very rare accident sequence. However, it can quantify the overall contribution of those accident sequences and identify dominant multi-unit sequences, whereas the conventional MCS method fails to quantify both contributions. In addition, the DQFM method can estimate importance measures of basic events with statistical error bounds. If one needs overall CDF and importance measures of basic events in MUPRA, the DQFM method is a good candidate for MUPRA quantification.

4. CONCLUSION

This paper proposes the hybrid approach for an MUPRA model using the DQFM method and MCSs. First, the approach utilizes the DQFM method to quantify a PRA model to estimate an overall CDF and dominant accident sequences with their occurrence frequencies. In addition, the approach also estimates a truncation value for MCSs. One can apply an automated algorithm to generate MCSs if one needs MCSs. In addition, this paper proposes a simple algorithm to generate importance measures of basic events.

References

[1] Division of Safety Systems, Nuclear Reactor Regulation, U.S Nuclear Regulatory Commission, Risk-Informed Improvements to the Standard Technical Specifications, September 2013, https://www.nrc.gov/docs/ML1328/ML13280A630.pdf, accessed 2024/05/20.
[2] International Atomic Energy Agency, Multi-unit Probabilistic Safety Assessment, Safety Report Series No. 110, IAEA, Vienna, (2023).
[3] Watanabe, Y., Oikawa, T., and Muramatsu, K., Development of the DQFM method to consider the effect of correlation of component failures in seismic PSA of nuclear power plant, Reliability Engineering and System Safety, Vol.79, No.3 (2003), pp.265–279.
[4] Shota Soga, Eishiro Higo, Hiromichi Miura, Proposal to use a DQFM result to set a truncation value for minimal cutsets [in Japanese], Spring Annual Meeting, 2D01, Atomic Energy Society of Japan
[5] Muhlheim, M. D., et al., "Initiating Events for Multi-Reactor Plant Sites," ORNL/TM-2014/533, 2014.
[6] Schroer, S. and Modarres, M., "An Event Classification Schema for Evaluating Site Risk in a Multi-unit Nuclear Power Plant Probabilistic Risk Assessment," Reliability Engineering and System Safety, 117 (2013): 40–51.
[7] Kenneth, K., et al., "A Framework for Addressing Site Integrated Risk," International Topical Meeting on Probabilistic Safety Assessment and Analysis, PSA 2015, April 2015, Sun Valley, USA, 2015.
[8] Hiromichi Miura, Shota Soga, Eishiro Higo, Yukihiro Higo, Development of Methods for Internal event (level 1) Multi-unit PRA, CRIEPI Report NR22003, April 2023