Enhancement of the FTeMC software for fault tree top event probability evaluation using Monte Carlo approach

Sang Hoon HAN*
Korea Atomic Energy Research Institute, 111, Daedeok-daero 989 Beon-gil, Yuseong-gu, Daejeon, 305-353, Korea
**Corresponding author: shhan2@kaeri.re.kr*

1. Introduction

The Monte Carlo method [1] can be used as a supplement to the minimal cut set approach when quantifying a fault tree, especially if minimal cut sets cannot be generated or if the results are in doubt.

KAERI (Korea Atomic Energy Research Institute) has developed software called FTeMC (Fault Tree top event probability Evaluation using Monte Carlo simulation) [2].

The Monte Carlo approach requires a lot of calculations for a large number of samples, and thus it can take a long time. In particular, the fault tree model of the domestic PSA (Probabilistic Safety Assessment) includes circular logic. In this case, the Monte Carlo method becomes more complicated and takes a long time to calculate.

This paper summarizes a study for reducing the computation time for a fault tree with circular logic when using the Monte Carlo method.

In Section 2, we explain the basic method of calculating the top event probability of a fault tree using the Monte Carlo method. In Section 3, we describe algorithms for reducing the calculation time. Section 4 presents the calculation time for several selected fault trees and conclusions.

2. Basic algorithm for Monte Carlo method

The Monte Carlo method for a fault tree analysis is a method for estimating the probability by counting how many times a failure has occurred during a number of trials. The top event probability estimation of a fault tree using the Monte Carlo method can be performed as follows [3].

- Repeat the following for N samples
  - Determine the state (True or False) of each basic event randomly
  - Determine the state (True or False) of the top event using the states of basic events
- If F failures occurred during N trials, the top event probability is evaluated as F / N

3. Algorithms to save the calculation time

The Monte Carlo method repeats the same calculation for a large number of trials. Small changes in recurring items can have a significant impact on the calculation time.

After reviewing and testing the algorithm of the FTeMC software, we have found that the following items have a large impact on the calculation time;

- Randomly determine the state of each basic event
  - Dagger sampling [1] is effective. It is implemented in the FTeMC.
- Calculate the state (failure or success) of the top event
  - Minimization of the number of calculations when calculating the state for each gate
  - Data array optimization and ordering of child in a gate
  - Fault tree simplification technique which is also used for minimal cut set generation
  - More calculation time is required for a fault tree containing circular logic. A new top-down iteration approach is developed [6].

A description of each item is shown below.

3.1 Fault tree simplification

Simplification of a fault tree greatly affects the calculation time for the minimum cut sets [4, 5]. It also affects the calculation time for the Monte Carlo method. The following features are reviewed and most of them are implemented in the FTeMC. However, the Shannon decomposition does not apply to the FTeMC because of its effectiveness compared to algorithm development efforts.

Merge gate

- It combines the same type of gates. As shown in the figure below, G2 and G3 are the same OR gates, and thus are combined into one gate.
- It can reduce the number of gates to be calculated

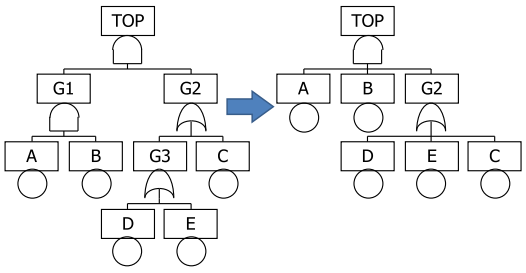


Fig. 1. Merge gate

Remove duplicates

- It deletes the duplicated child in a gate.
- This can be caused by the process of simplifying the fault tree or by a user's mistake.

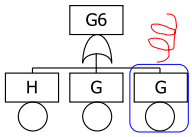


Fig. 2. Remove duplicates

Modularization

- It replaces an independent subtree or a module with a basic event.
- It reduces the size of the fault tree.

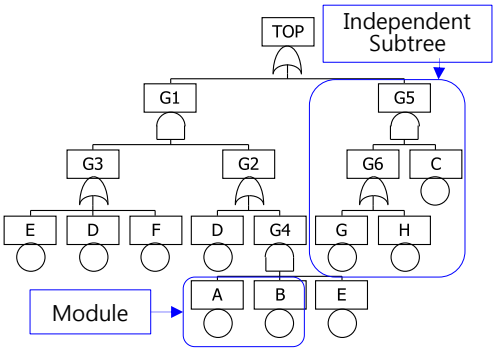


Fig. 3. Fault tree modularization

Shannon decomposition

- This is a powerful feature for fault tree simplification.
- It does not apply to the FTeMC because it is not as effective as algorithm development efforts.
- It can simplify a fault tree for a case in which T(r=1) = 1 as follows:
  - T = r * T(r=1) + /r * T(r=0) = r + T(r=0)
- An example of Shannon decomposition is shown below.

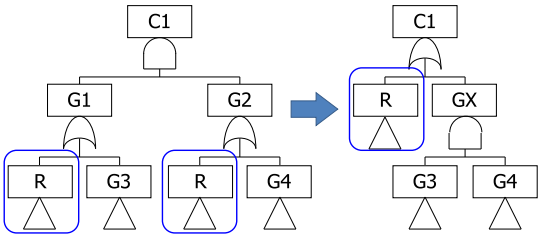


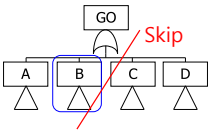
Fig. 4. Shannon Decomposition

3.2 Optimization of fault tree structure

It is necessary to check the state of each gate to calculate the state of the top event. The following features are implemented into the FTeMC.

Skip unnecessary children

- When calculating the state of each gate using the states of its children, unnecessary children are skipped without checking once the state of the gate is determined.
- An OR gate is True if any child is True. When checking the states of the children, if anyone is true, the rest of the children do not need to be examined. A similar approach can be used for a AND gate.
- It can reduce the number of tests in each gate.

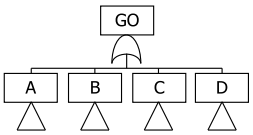



```
Sg = False // initialize State of Gate Sg
For each Ci  // Ci = i-th child
    If Si is True // Si = state of Ci
        Sg = TRUE
        Exit For
```

Fig. 5. Skip unnecessary children

Ordering of children

- It sorts the children in descending order of probability in an OR gate, and in ascending order of probability in a AND gate.
- It is recommended to estimate the probability of each child heuristically.
- It arranges basic events first rather than gates.
- It can reduce the number of tests in each gate



P(A) > P(B) > P(C) > P(D)

Fig. 6. Ordering of children

Restructuring of the fault tree array

- The FTeMC examines a fault tree with a top-down depth-first search. It recreates the entire fault tree structure in this order.
- It removes unused gates and basic events.
- It is for efficient memory access
- The gates are stored in a fault tree array in the order of accessing the fault tree.
- For the example shown in the figure below, the order will be G4, G2, G3, G1, G6, G5 and TOP.

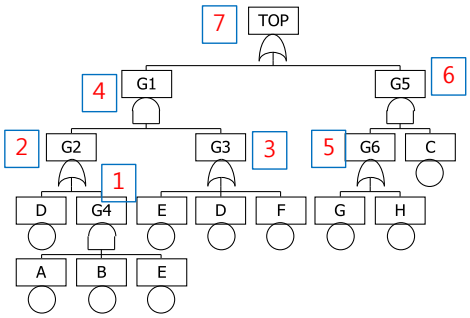


Fig. 7. Restructuring of fault tree array

3.3 Top-down iteration approach

The FTeMC has implemented several approaches to calculate the state of the top event. The characteristics of the approaches are as follows;

- Top-down approach: Once the state of the gate is determined, it is efficient to skip the remaining children without checking them. However, it cannot be applied to a fault tree with circular logic.
- Top-down circular approach: It can be used for a fault tree with circular logic [3]. When a specific gate is involved in circular logic, the gate can have different results depending on the location, and thus it requires repeated calculations which takes a long time.
- Iteration approach: it is an efficient approach for a fault tree with circular logic. However, it has a disadvantage in calculating the state of all gates. Note that top-down approach does not need to calculate the states of all gates.

A new algorithm called a top-down iteration approach [6] was recently developed.

- The top-down circular approach is modified to not recalculate the gate once the state is calculated. The modified approach produces the same result as the iteration approach. However, the children can be skipped once the state of a gate is determined. The calculation for the states of the gates are repeated until the results converge for fault trees with circular logic.
- It is a way to combine the advantages of top-down circular and iteration approaches.
- This is an effective method for a fault tree with circular logic, and does not degrade the performance much better than a top-down approach for a fault tree without circular logic.

The algorithm is illustrated in Figs. 8 through 10.

```
// Main iteration part
S() for Gates = False // Initialize Vector S = False for Gate
Do   // Repeat until converged
  Converged = True  // Initialize a variable for convergence checking
  C() = False          //  Initialize Vector C (Ci : True if i-th gate calculated)
  St = GetState(T)   //  Calculate the state of the top event, recursively
Until (Converged)
```

Fig. 8. Iteration part for top-down iteration approach

```
// Top-down function to calculate the state of g, Sg
Function GetState(g)
  If (g is a gate and Cg=False) then  // for a not-calculated gate
    Cg = True          // If Calculated, Cg = True
    s = CalculateGateState (g)  // Calculate the State of Gate g, Sg
    if (s ≠ Sg) then     // Check convergence
      Converged = False
    Sg = s
  Return Sg
```

Fig. 9. Function to calculate the state of a gate/event

```
// Sub-function to calculate the state of a Gate, Sg
Function CalculateGateState(g)
  If (g is a OR gate) then  // for a OR gate
    For each child j ∈ g  // for each child of g
      Sj = GetState (j)
      if (Sj ≡ True) then   // if any one child is True
        return True        //    Sg = True
    return False           // Otherwise, Sg = False
  … // For other type of gate
```

Fig. 10. Function to calculate the state of a gate

4. Application results and summary

The various features described in Section 3 have been applied to the FTeMC Version 2. Table 1 shows the test results for the selected fault trees. The fault tree models come from the PSA models for nuclear power plants.

- P1 and P2 are fault trees without circular logic.
- M1 and M2 are fault trees for multiple unit PSAs, and thus the size of the models are large. They also have circular logic.
- C1 and C2 are PSA models including multiple initiating events. The CDF (Core damage frequency) for C1 or C2 can be calculated by calculating CCDPs (Conditional core damage probability) for each initiating event and combining them, as shown in remark 7 in Table 1.

V1 represents FTeMC Version 1, and V2a, V2b and V2c represent FTeMC Version 2. The characteristics of each version are given in remarks 1 through 4. V2c uses all features developed in FTeMC Version 2.

On average, V2a is 2.1 times faster than V1, V2b is 2.2 times faster than V2a, and V2c is 3.6 times faster than V2b. Overall, FTeMC Version 2 is 3 to 50 times faster than Version 1.

All features implemented in FTeMC Version 2 appear to be effective, such as an optimization of the fault tree

structure, fault tree simplification, and the top-down iteration approach.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT: Ministry of Science and ICT) (No. 2017M2A8A4015287).

REFERENCES

[1] Kumamoto, H., 1980. Dagger-sampling Monte Carlo for system unavailability evaluation. IEEE Transactions on Reliability R-29 (2), 122–125 (1980)

[2] FTeMC Quick Guide - Fault Tree top event probability Evaluation using Monte Carlo simulation, KAERI, KAERI-ISA-Memo-FTeMC-01, Rev. 1, 2017

[3] Sang Hoon Han and Ho-Gon Lim, Top event probability evaluation of a fault tree having circular logics by using Monte Carlo method, Nuclear Engineering and Design, 243 (2012) 336-340

[4] Woo Sik Jung, Advanced Features and Performance Evolution of Fault Tree Quantifier FORTE, 5th Korea-Japan PSA Workshop, Seoul, Korea, 1999

[5] llkka Niemela, On simplification of large fault trees, Reliability Engineering and System Safety 44 (1994) 135-138

[6] A top-down iteration algorithm for Monte Carlo method for probability estimation of a fault tree with circular logic, to be published (2018)

Table 1. Summary of calculation time for selected fault trees

Case	V1[1] (sec)	V2a[2] (sec)	V2b[3] (sec)	V2c[4] (sec)	# of Samples	Model
P1[5]	31	14	8	8	10^6	No Circular Logic, 4530 Gates/1635 Basic Events
P2[5]	32	14	12	13	10^6	No Circular Logic, 3654 Gates / 2095 Basic Events
M1[6]	5350	1477	597	708	10^6	Circular Logic, 49231 Gates / 23734 Basic Events
M2[6]	3198	1495	410	65	10^6	Circular Logic, 37345 Gates / 21217 Basic Events
C1[6, 7]	857	790	335	39	10^6	Circular Logic, 4530 Gates / 1258 Basic Events 7 Initiating Events
C2[6, 7]	3013	2442	1537	378	10^6	Circular Logic, 5056 Gates / 2458 Basic Events 20 Initiating Events

1) V1 : FTeMC Version 1, w/o optimization of FT structure, w/o FT simplification
2) V2a : FTeMC Version 2, Optimization of FT structure, w/o FT simplification
3) V2b : FTeMC Version 2, Optimization of FT structure, FT simplification
4) V2c : FTeMC Version 2, Optimization of FT structure, FT simplification, Top-down iteration approach
5) P1 and P2 have no circular logic. Top-down approach is used for V1, V2a, V2b.
6) M1, M2, C1 and C2 have circular logics. Iteration approach is used for V1, V2a, V2b.
7) C1 and C2 have several initiating events. CDF is calculated by summing the result for each initiating event:

$$\text{CDF} = \sum_i (f(IE_i) * CCDP(IE_i = True\ ,\ IE_{j\neq i} = False))$$