*Proc. of the European Safety and Reliability Conference (ESREL 2026)*
*Edited by* José C. Matos, Paulo B. Lourenço, Michael Beer, Daniel Oliveira,
Edoardo Patelli, Hélder S. Sousa, Rui A. Silva and Monica Santamaria
 *Published by* Research Publishing, Singapore.
doi: 10.3850/ESREL2026061419_esrel26-p26155-cd

# Culling methods for minimal cut sets including dependencies

Silvia Tolo

*Resilience Engineering Research Group, University of Nottingham, U.K.*
*E-mail: silvia.tolo@nottingham.ac.uk*

John Andrews

*Resilience Engineering Research Group, University of Nottingham, U.K.*
*E-mail: John.Andrews@nottingham.ac.uk*

Binary Decision Diagrams (BDDs) are widely used in reliability analysis as an efficient representation of the Boolean logic underlying Fault and Event Tree models. More recently, BDDs have become a core component of the Dynamic and Dependent Tree Theory ($D^2T^2$), where they enable the inclusion of dependencies within traditional Fault/Event Tree frameworks.
Despite their advantages, the construction of BDDs from large tree models can be challenging and, in some cases, infeasible. Although several mitigation strategies have been proposed, existing approaches do not account for dependencies, limiting their applicability to $D^2T^2$ analyses.
This study proposes novel algorithms for identifying and culling Fault Tree minimal cut sets and for constructing BDDs from them while explicitly accounting for dependencies between basic events. By integrating culling criteria based on order, probability, and frequency within a framework that explicitly accounts for event dependencies, the proposed approach enables the generation of BDDs regardless of model complexity, thereby ensuring the applicability of $D^2T^2$ to large and complex Fault Tree models.

*Keywords*: Binary Decision Diagrams, Dynamic and Dependent Tree Theory, Fault Trees, Minimal Cut Sets, Dependencies, Culling.

## 1. Introduction

Dynamic and Dependent Tree Theory ($D^2T^2$) extends conventional fault and event tree analysis by enabling the explicit modelling of dependencies and joint probabilistic behaviour among traditional Fault Tree (FT) basic and intermediate events (Tolo and Andrews, 2025; Andrews and Tolo, 2023). The approach relies on Binary Decision Diagrams (BDDs) to evaluate system reliability metrics through dedicated structure-based algorithms (Tolo and Andrews, 2023, 2022). As a result, the availability of a BDD representation is a prerequisite for the application of the $D^2T^2$ framework.
BDDs were introduced in reliability analysis as an efficient alternative to classical FT quantification since, when directly constructed from the FT structure, they allow the exact computation of top event metrics (Rauzy, 2008). However, the FT to BDD conversion process is characterised by exponential worst case complexity, which can make BDD construction infeasible for large or highly interconnected models (Lopuhaä-Zwakenberg, 2025).
To address this limitation, BDDs can be constructed from Minimal Cut Sets (MCSs) extracted from the original FT. MCSs represent minimal combinations of basic events leading to the top event and can be simplified through culling procedures based on order or probabilistic relevance (Sinnamon and Andrews, 1996; Choi and Cho, 2005). While this approach introduces approximations and prevents exact quantification, it preserves the structural advantages of BDDs and ensures the applicability of $D^2T^2$ to large-scale systems.
While approximation techniques have been introduced within the $D^2T^2$ framework (Yan and Andrews, 2024), existing MCS culling methods largely rely on independence assumptions and do not explicitly preserve dependency structures. This work proposes algorithms for MCS iden-

tification and culling by order, probability, and frequency that explicitly account for dependencies, enabling the construction of reduced BDDs consistent with joint probabilistic modelling and extending the applicability of $D^2T^2$ to large and complex fault trees.

## 2. MCS identification

**Algorithm 1:** *expand* function

**Input:** $g_i$
**Output:** $\mathbf{CS^i} = \{\mathbf{cs^i_1}, \mathbf{cs^i_2}, ..., \mathbf{cs^i_h}\}$
% gate $g_i$ cut sets
($t_i$, $\mathbf{S^i}$)=gate($g_i$)
% $\mathbf{S^i}$ array of $i$ gate input
**if** $t_i = 0$ **then**
  $\mathbf{cs^i_0} \leftarrow \mathbf{S^i}$
  append $\mathbf{cs^i_0}$ to $\mathbf{CS^i}$
**else if** $t_i = 0$ **then**
  **for** *each* $s^i_j \in \mathbf{S^i}$ **do**
    $\mathbf{cs^i_j} \leftarrow s^i_j$
    append $\mathbf{cs^i_j}$ to $\mathbf{CS^i}$
  **end**
**end**

MCS identification expands gates into their basic event combinations according to logic, progressively replacing intermediate events until the top gate is expressed solely in basic events. These combinations, called gate cut sets, represent event sets triggering the gate. Top gate cut sets correspond to system level cut sets, which are minimized to extract MCSs. Algorithm 1 describes the expansion of each gate ($g_i$) based on its logic type $t_i$ ($t_i = 0$ for AND, $t_i = 1$ for OR).
The search can proceed top-down (from the top event) or bottom-up (from terminal gates). This study adopts a bottom-up approach, starting from gates whose inputs are only basic events (i.e., terminal gates), which can be expanded directly. For instance, let's consider the OR gate labelled $g_{16}$ in Fig.1. This yields two cut sets:

$$\begin{aligned} \mathbf{cs^{g16}_1} &= \{X_{16}\} \\ \mathbf{cs^{g16}_2} &= \{X_{17}\} \end{aligned} \tag{1}$$

**Algorithm 2:** *substitute* function

**Input:** $g_i, g_k \mid g_k \in \mathbf{S^i}$
**Output:** $\mathbf{CS^i} \mid g_k \notin \mathbf{cs^i_h} \forall h$
define: **newCS**
% empty set data structure
**if** $\mathbf{CS^i}$ *is empty* **then**
  $\mathbf{CS^i} \leftarrow \underline{expand}(g_i)$
**else**
  $\mathbf{CS^i} = cutsets(g_i)$
  % array of $n$ cut sets
  $\mathbf{CS^k} = cutsets(g_k)$
  % array of $m$ cut sets
  **for** *each* $\mathbf{cs^i_j} \in \mathbf{CS^i}$ **do**
    **if** $g_k \in \mathbf{cs^i_j}$ **then**
      delete $g_k$ from $\mathbf{cs^i_j}$
      **for** *each* $\mathbf{cs^k_x} \in \mathbf{CS^k}$ **do**
        $\mathbf{newCS} \leftarrow \mathbf{cs^i_j}$
        append $\mathbf{cs^k_x}$ to **newCS**
        append **newCS** to $\mathbf{CS^i}$
      **end**
      delete $\mathbf{cs^i_j}$ from $\mathbf{CS^i}$
  **end**
**end**

while the AND gate $g_8$ from the same model yields one:

$$\mathbf{cs^{gs}_1} = \{x_7, x_8\} \tag{2}$$

Non-terminal gates require the substitution of intermediate events with their cut sets, iteratively propagating upward until the top event is expressed in basic events. Details of the substitution procedure are provided in Algorithm 2. Once the top event is fully expanded, non-minimal cut sets are removed to obtain MCSs.

### 2.1. *Culling by order*

As noted in Section 1, real-world applications often require reducing model complexity by truncating the list of MCSs considered in the analysis. This truncation can be based on MCS order, retaining only those with fewer events than a specified threshold. To implement this, the expansion and substitution procedures (Algorithms 1 and 2) can be modified to discard non-compliant cut sets

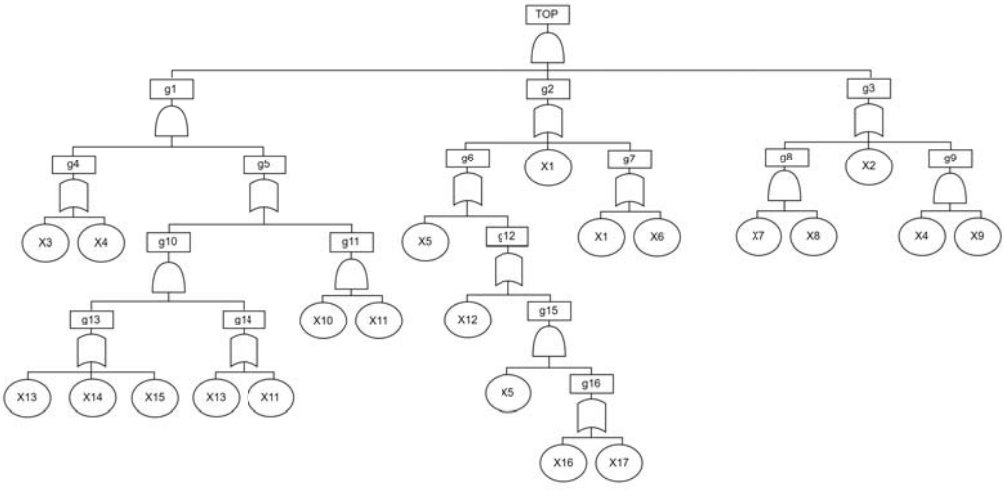



Fig. 1.: FT example model (Reay and Andrews, 2002)

before full factorization, improving computational efficiency. Given an upper bound $orderUB$ for MCS order, line 4 of Algorithm 1 can be replaced with:

**if** $length(\mathbf{cs_0^i}) \leq orderUB$ **then**
  $\mathbf{CS^i}[0] \leftarrow \mathbf{cs_0^i}$

Similarly, line 14 of Algorithm 2 becomes:

**if** $length(\mathbf{newCS}) \leq orderUB$ **then**
  append $\mathbf{newCS}$ to $\mathbf{CS^i}$

### 2.2. *Culling by probability*

Under the independence assumption, the probability of a MCS equals the product of its basic event probabilities. This makes probability tracking during MCS identification straightforward and allows early elimination of cut sets below a predefined threshold, improving computational efficiency. To compute probabilities concurrently with MCS identification, a parallel data structure is introduced:

$$\mathbf{P} = \mathbf{P^1}, \mathbf{P^2}, \ldots, \mathbf{P^n} \tag{3}$$

where $\mathbf{P^i} = p_1^i, p_2^i, \ldots, p_m^i$ stores probabilities for the $m$ cut sets of gate $g_i$. Each $p_j^i$ represents the probability of cut set $\mathbf{cs_j^i}$ (complete if fully expanded, partial otherwise). As for culling by order, the MCS identification algorithms must be modified accordingly. For a lower bound $probabilityLB$, line 4 of Algorithm 1 becomes:

**for** *each* $s_j^i \in \mathbf{S^i} \mid s_j^i \in \mathbf{BE}$ **do**
  $p_0^i \leftarrow p_0^i \cdot q(s_j^i)$
**if** $p_0^i \geq probabilityLB$ **then**
  append $\mathbf{cs_0^i}$ to $\mathbf{CS^i}$ append $p_0^i$ to $\mathbf{P^i}$

For OR gates, line 8 is replaced with:

**if** $s_j^i \in \mathbf{BE}$ ***and*** $q(s_j^i) \geq probabilityLB$ **then**
  append $\mathbf{cs_j^i}$ to $\mathbf{CS^i}$ append $q(s_j^i)$ to $\mathbf{P^i}$

Probability propagation during substitution replaces lines 12–13 in Algorithm 2 with:

$p^{\text{intersection}} \leftarrow \prod q(i_l)$ for repeated events
$newP \leftarrow p_j^i \cdot \frac{p_x^k}{p^{\text{intersection}}}$ **if** $newP \geq probabilityLB$ **then**
  append $\mathbf{newCS}$ and $newP$ to $\mathbf{CS^i}$, $\mathbf{P^i}$

These changes ensure only cut sets above the probability threshold are retained, regardless of full expansion, focusing the analysis on the most significant MCSs.

### 2.3. *Culling by frequency*

Each basic event in a Minimal Cut Set (MCS) can be classified according to its role in the failure mechanism as either an initiating or an enabling event. Initiating events trigger the fail-

**Algorithm 3:** Cut set identification

**Output:** $\mathbf{CS^{TOP}} = \{\mathbf{cs_1^{TOP}}, \mathbf{cs_2^{TOP}}, ..., \mathbf{cs_l^{TOP}}\} \mid \mathbf{cs_k^{TOP}} \subseteq \mathbf{BE} \forall k = 1, ..., l$
% top event cut sets
define: $\mathbf{ToProcess} \leftarrow \{\}$
% empty array of gate identifiers
**for** *each* $\mathbf{G_i} \in \mathbf{G} \mid S_i \subseteq \mathbf{BE}$ **do**
  $\mathbf{CS}(g_i) \leftarrow \underline{expand}(g_i)$
  % see expansion in Algorithm 1
  $\mathbf{ToProcess}[end+1] \leftarrow g_i$
**end**
**repeat**
  $p = \mathbf{toProcess}[1]$
  % first element of the array
  $\mathbf{Pa} = parents\{p\}$
  % gates including p as input
  **for** *each* $pa_i$ *in* $\mathbf{Pa}$ **do**
    $\mathbf{CS^{pa_i}}$=substitute($p$,$pa_i$)
    % see substitution in Algorithm 2
    **if** $cs_g^{pa_i} \subseteq \mathbf{BE} \forall g$ **then**
      $\mathbf{ToProcess}[end+1] \leftarrow pa_i$
  **end**
  delete $\mathbf{toProcess}[0]$ from $\mathbf{toProcess}$
**until** $\mathbf{ToProcess}[0] = TOP$

**Algorithm 4:** Cut sets minimization

**Output:** $\mathbf{MCSs} = \{\mathbf{mcs_1}, \mathbf{mcs_2}, ..., \mathbf{mcs_f}\}$
% array of f minimal cut sets
define: $\mathbf{CS^{TOP,i}} \subseteq \mathbf{CS^{TOP}} \mid length(\mathbf{cs_j^{TOP}}) = i, \forall \mathbf{cs_j^{TOP}} \in \mathbf{CS^{TOP,i}}$
define: $maxOrder \geq length(\mathbf{cs_j^{TOP}}) \forall \mathbf{cs_j^{TOP}} \in \mathbf{CS^{TOP}}$
**for** *i from 1 to maxOrder* **do**
  **for** *each* $\mathbf{cs_j^{TOP}} \in \mathbf{CS^{TOP,i}}$ **do**
    **if** $\mathbf{cs_j^{TOP}} \subseteq \mathbf{cs_k^{TOP}} \mid \mathbf{cs_k^{TOP}} \in \mathbf{CS^{TOP,l}}$, *where* $l \geq i$ **then**
      delete $\mathbf{cs_k^{TOP}}$ from $\mathbf{CS^{TOP}}$
  **end**
**end**
$\mathbf{MCSs} \leftarrow \mathbf{CS^{TOP}}$

ure sequence, while enabling events must already have occurred for the top event to materialize. This role is context dependent rather than intrinsic to the event. For example, in a two pump system, the first pump to fail acts as the initiator and the second as the enabler, regardless of their identity. The computation of MCS frequency therefore requires identifying the implicit failure sequences associated with the MCS, defined as the admissible combinations of initiating and enabling events. The MCS frequency is obtained by summing the frequencies of these sequences and can be expressed as:

$$w(\mathbf{mcs}i) = \sum j \mid s_j \in \mathbf{I} w(s_j) \prod_{k|k \neq j} q(s_k), \tag{4}$$

where $w(s_j)$ is the frequency of an initiating event $s_j \in \mathbf{I}$ and $q(s_k)$ is the probability of the remaining enabling events $s_k \in \mathbf{E}$. Because frequency evaluation depends on sequence enumeration, culling by frequency cannot be performed during MCS identification and thus provides no computational savings at that stage. It remains, however, useful for constructing approximate BDDs when exact solutions are infeasible. After identifying the set $\mathbf{MCS} = \mathbf{mcs}_1, \ldots, \mathbf{mcs}_r$ as shown in Algorithm 4, frequency culling can be applied using a threshold $frequencyUB$:

**for** *each* $\mathbf{mcs}i \in \mathbf{MCS}$ **do**
  **if** $w\mathrm{mcs}_i < frequencyUB$ **then**
    remove $\mathbf{mcs}_i$

For convenience, the MCS frequency is computed as:

**for** *each* $s_j \in \mathbf{mcs}i \mid s_j \in \mathbf{I}$ **do**
  $w\mathrm{mcs}i \leftarrow w\mathrm{mcs}i + \dfrac{w(s_j)}{q(s_j)} \cdot q\mathrm{mcs}i$

Here, $w(s_j)$ and $q(s_j)$ are input parameters, while $q\mathrm{mcs}_i$ denotes the MCS probability already tracked during cut set processing (Section 2.2). Only minor adjustments to Algorithm 4 are required to retain these probability associations.

## 3. Extension to $D^2T^2$

The $D^2T^2$ methodology accounts for dependencies among basic events by using joint proba-

bilities instead of individual event probabilities, which must therefore be used when evaluating MCS metrics. While MCS order remains unaffected (see Section 2.1) and frequency culling follows the same procedure as under independence (Section 2.3), probability-based culling must be adapted to explicitly handle joint probabilistic terms, as discussed next.

### 3.1. *Probability culling with dependencies*

Under the assumption of independence, adding a new event during MCS identification always decreases the overall probability, as it involves multiplying by a value less than 1. With dependencies this no longer holds, since adding a dependent event can increase the probability through joint terms in the same dependency group. Even so, probability culling can still be used concurrently with MCS identification by considering only independent events during identification. Any combination that includes dependent events will still give a probability below 1, so the overall value continues to decrease. This is achieved modifying the expansion procedure in Algorithm 1 adding at line 4 the pseudo code:

**for** *each* $s_j^i \in \mathbf{S^i} \mid s_j^i \in \mathbf{BE}$ **do**
  **if** $s_j^i \in \mathbf{DG^0}$ **then**
    $p_0^i \leftarrow p_0^i \cdot q(s_j^i)$
**if** $p_0^i \geq probabilityLB$ **then**
  append $\mathbf{cs_0^i}$ to $\mathbf{CS^i}$ append $p_0^i$ to $\mathbf{P^i}$

where $\mathbf{DG^0}$ is the set of FT independent events. OR gate expansion is obtained adding at line 8 of the same algorithm:

**if** $s^i j \in \mathbf{DG^0}$ *and* $q(s^i j) \geq probabilityLB$ **then**
  $\mathbf{cs_j^i} \leftarrow s^i j$ append $q(s^i j)$ to $\mathbf{P^i}$ append $\mathbf{cs_j^i}$ to $\mathbf{CS^i}$

The substitution function follows the same idea as in Section 2.2. These updates allow early removal of cut sets that fall below the threshold when only independent events are considered. This estimate is still incomplete, since dependent events may later lower the probability further. The minimization step must therefore check final probabilities after dependencies are included.

## 4. BDD construction from MCSs

BDDs are directed acyclic graphs where nodes encode if-then-else (ITE) statements. A generic internal node $\mathbf{X_i}$ is defined as:

$$\mathbf{X_i} = ite(x_i, \mathbf{X_j}, \mathbf{X_k}) \tag{5}$$

where $x_i$ is the node label (typically a basic event), and $\mathbf{X_j}$ and $\mathbf{X_k}$ are the nodes on the 1-branch and 0-branch, corresponding to the occurrence and non-occurrence of $x_i$. Moving from the top to a terminal vertex (1 or 0) through these branches yields event sequences: paths ending in 1 validate the Boolean function (e.g., top event or MCS occurrence), while those ending in 0 negate it. For large or complex FT models, direct conversion to BDDs may be infeasible. In such cases, BDDs can be constructed from identified MCSs (wether exact or culled) using two steps:

- build a BDD for each MCS of the FT
- combine them into a single BDD for the top event

Since events in a MCS are joined by AND logic, its BDD is straightforward: each event lies on the 1-branch of the previous node. Algorithm 5 formalizes this process. After all MCS BDDs are created, they are joined through repeated OR operations to obtain the top event BDD (Algorithm 6). For details on BDD logic operations, see (Rauzy, 1993).

**Algorithm 5:** Building the BDD of $\mathbf{MCS^j}$

**Output:** $\mathbf{X^j} = \{\mathbf{X_1^j}, \mathbf{X_2^j}, ..., \mathbf{X_h^j}\}$ % nodes ite structures of $\mathbf{MCS^j}$ BDD
define: $\mathbf{MCS^j} = \{x_1^j, x_2^j, ..., x_h^j\}$
% where $x_n^j$ is the identifier of the n-th basic event in $\mathbf{MCS^j}$
$\mathbf{X_{h+1}^j} = 1$
**for** *each* $\mathbf{x_i^j}$ *in* $\mathbf{MCS^j}$ **do**
  $\mathbf{X_i^j} \leftarrow ite(x_i^j, \mathbf{X_{i+1}^j}, 0)$
**end**

**Algorithm 6:** From MCSs to top event BDD

**Output:** $\mathbf{Z} = \{\mathbf{Z_1}, \mathbf{Z_2}, ..., \mathbf{Z_n}\}$

`% final BDD nodes ite structures`

define:$\mathbf{X} = \{\mathbf{X^1}, \mathbf{X^2}, ..., \mathbf{X^k}\}$

`% see Algorithm 5 for` $\mathbf{X^j}$

$\mathbf{Z} \leftarrow \mathbf{X^1}$

remove $\mathbf{X^1}$ from $\mathbf{X}$

**for** *each* $\mathbf{X^j}$ *in* $\mathbf{X}$ **do**

  $\mathbf{Z} \leftarrow \mathbf{Z} \vee \mathbf{X^j}$  `% ∨ denotes BDD OR`

**end**

## 5. Numerical Application

The proposed approach was applied to the FT model in Figure 1 using the input reliability data shown in Table 1. The identification procedure in Section 2 produced 80 MCSs, with orders from 4 to 6, probabilities between $1 \cdot 10^{-13}$ and $7 \cdot 10^{-09}$, and frequencies between $1 \cdot 10^{-15}$ and $9 \cdot 10^{-11}$ $h^{-1}$.

Table 1.: Reliability data for basic event in Fig.1.

| Basic Event | Failure Probability | Failure Frequency |
|---|---|---|
| X1 | $3.00 \cdot 10^{-3}$ | $1.94 \cdot 10^{-4}$ |
| X2 | $4.50 \cdot 10^{-3}$ | $9.90 \cdot 10^{-7}$ |
| X3 | $8.00 \cdot 10^{-3}$ | $2.15 \cdot 10^{-6}$ |
| X4 | $1.00 \cdot 10^{-2}$ | $1.37 \cdot 10^{-5}$ |
| X5 | $3.50 \cdot 10^{-3}$ | $3.92 \cdot 10^{-6}$ |
| X6 | $2.50 \cdot 10^{-3}$ | $8.50 \cdot 10^{-7}$ |
| X7 | $1.50 \cdot 10^{-2}$ | $2.44 \cdot 10^{-6}$ |
| X8 | $1.20 \cdot 10^{-2}$ | $6.40 \cdot 10^{-7}$ |
| X9 | $9.00 \cdot 10^{-3}$ | $2.27 \cdot 10^{-6}$ |
| X10 | $4.00 \cdot 10^{-3}$ | $3.92 \cdot 10^{-6}$ |
| X11 | $7.00 \cdot 10^{-3}$ | $6.22 \cdot 10^{-5}$ |
| X12 | $1.50 \cdot 10^{-2}$ | $8.76 \cdot 10^{-6}$ |
| X13 | $5.00 \cdot 10^{-3}$ | $4.86 \cdot 10^{-6}$ |
| X14 | $8.00 \cdot 10^{-3}$ | $1.12 \cdot 10^{-4}$ |
| X15 | $6.50 \cdot 10^{-3}$ | $9.90 \cdot 10^{-7}$ |
| X16 | $1.20 \cdot 10^{-2}$ | $3.53 \cdot 10^{-5}$ |
| X17 | $6.00 \cdot 10^{-3}$ | $7.86 \cdot 10^{-6}$ |

The analytical solution yielded a top event probability of $2.12 \cdot 10^{-08}$ and frequency of $2.38 \cdot 10^{-10}, h^{-1}$. The culling strategies in Section 2 were applied, resulting in three simplified models:

- By order: MCSs with order $\leq 4$ (12 of 80 MCSs)
- By probability: MCSs with probability $\geq 10^{-10}$ (14 MCSs)
- By frequency: MCSs with frequency $\geq 10^{-11} h^{-1}$ (5 MCSs)

Figure 2 compares the resulting BDDs, while Table 2 summarizes the top event metrics for each case.

Table 2.: FT analysis results by culling strategy.

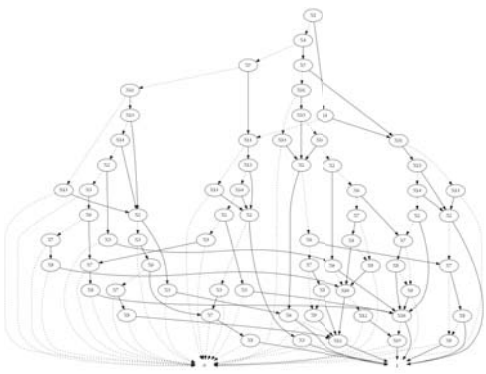
| Culling Strategy | Failure Probability | Failure Frequency |
|---|---|---|
| None | $2.12 \cdot 10^{-8}$ | $2.38 \cdot 10^{-10}$ |
| Order ($\leq 4$) | $2.03 \cdot 10^{-8}$ | $2.21 \cdot 10^{-10}$ |
| Probability ($\geq 10^{-10}$) | $2.05 \cdot 10^{-8}$ | $2.21 \cdot 10^{-10}$ |
| Frequency ($\geq 10^{-11}$) | $1.26 \cdot 10^{-8}$ | $2.01 \cdot 10^{-10}$ |

Order and probability culling introduce minor errors ($\leq 4\%$ and $\leq 3\%$ respectively), while frequency culling, due to its stricter threshold, shows larger deviations ($\approx 40\%$ for probability and $15\%$ for frequency).

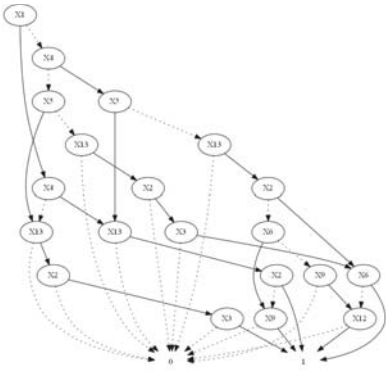
The same threshold values were adopted in the analysis of the $D^2T^2$ model obtained introducing in the FT two dependency groups: $\mathbf{DG1} = X1, X2$ and $\mathbf{DG2} = X4, X5$. The joint reliability metrics characterising these groups are shown in Table 3.

Table 3.: DG1 and DG2 joint reliability metrics.

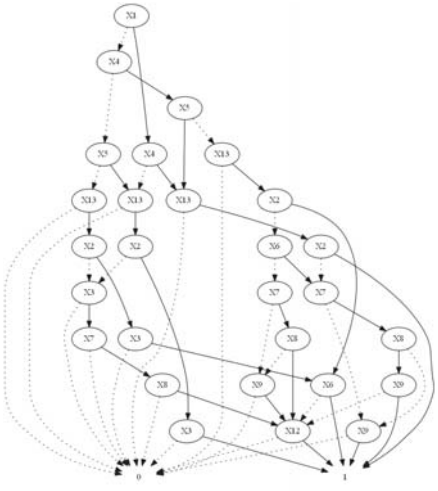
| Joint States | Probability | Intensity |
|---|---|---|
| $X1, X2$ | $1.6205 \cdot 10^{-1}$ | $8.4478 \cdot 10^{-1}$ |
| $\overline{X1}, X2$ | $2.1658 \cdot 10^{-1}$ | $2.7614 \cdot 10^{-1}$ |
| $X1, \overline{X2}$ | $2.1612 \cdot 10^{-1}$ | $2.7478 \cdot 10^{-1}$ |
| $\overline{X1}, \overline{X2}$ | $4.0525 \cdot 10^{-1}$ | $6.7541 \cdot 10^{-1}$ |
| $X4, X5$ | $1.1765 \cdot 10^{-2}$ | $7.0588 \cdot 10^{-2}$ |
| $\overline{X4}, X5$ | $3.5294 \cdot 10^{-2}$ | $4.5882 \cdot 10^{-1}$ |
| $X4, \overline{X5}$ | $3.5294 \cdot 10^{-2}$ | $4.5882 \cdot 10^{-1}$ |
| $\overline{X4}, \overline{X5}$ | $9.1765 \cdot 10^{-1}$ | $9.1765 \cdot 10^{-1}$ |



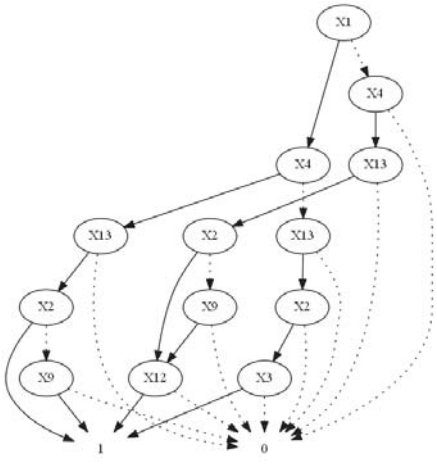
(a) None



(b) By order



(c) By probability




(d) By frequency

Fig. 2.: BDDs obtained applying culling methods assuming independence

As shown in Table 4, accounting for dependencies increases the top event reliability metrics by over an order of magnitude across all analytical and culling strategies when compared with the independence assumption.

The same 12 cut sets and corresponding BDD structure are obtained when culling the $D^2T^2$ model by order, as dependencies do not affect MCS order. Nevertheless, the associated probabilities and frequencies differ when dependency groups **DG1** and **DG2** are considered, leading to the top event metrics reported in Table 4.

Probabilistic based culling yields different MCS selections despite identical thresholds, retaining 42 and 72 of the original 80 MCSs when

Table 4.: $D^2T^2$ analysis output by culling strategy.

| Culling Strategy | Failure Probability | Failure Frequency |
|---|---|---|
| None | $3.59 \cdot 10^{-07}$ | $7.53 \cdot 10^{-07}$ |
| Order ($\leq 4$) | $3.49 \cdot 10^{-07}$ | $7.31 \cdot 10^{-07}$ |
| Probability ($\geq 10^{-10}$) | $3.59 \cdot 10^{-07}$ | $7.52 \cdot 10^{-07}$ |
| Frequency ($\geq 10^{-11}$) | $3.59 \cdot 10^{-07}$ | $7.53 \cdot 10^{-07}$ |

culling by probability and frequency, respectively. The larger number of MCSs included results in improved accuracy, while the approximate solutions remain close to the exact values and within the same order of magnitude in all cases. The maximum absolute error observed is $8.6 \cdot 10^{-08}$ for the top event probability obtained when culling by frequency.

Overall, the trade off between accuracy and computational complexity can be further optimised by tuning the culling parameters according to the specific analysis requirements.

## 6. Conclusions

This study introduces novel algorithms for the identification and culling of Fault Tree minimal cut sets by order, probability, and frequency, explicitly accounting for dependencies among basic events to support applications within the Dynamic and Dependent Tree Theory ($D^2T^2$) framework.

By reducing the complexity of the original Fault Tree models, the proposed methods enable the construction of Binary Decision Diagrams even for large scale systems, thereby extending the applicability of $D^2T^2$ to reliability problems of arbitrary size and complexity.

Although culling introduces approximations that may lead to an underestimation of system failure probability, the approach remains informative and allows analysts to tune the culling parameters to achieve a desired trade off between accuracy and computational efficiency within $D^2T^2$ analyses.

#### Acknowledgement

This work is funded by the Lloyd's Register Foundation, an independent global charity that helps to protect life and property at sea, on land, and in the air, by supporting high quality research, accelerating technology to application and through education and public outreach.

## References

Andrews, J. and S. Tolo (2023). Dynamic and dependent tree theory (d2t2): A framework for the analysis of fault trees with dependent basic events. *Reliability Engineering & System Safety 230*, 108959.

Choi, J. S. and N. Z. Cho (2005). Truncation error evaluation method for minimal cut set-based fault tree analysis. *Journal of nuclear science and technology 42*(10), 854–860.

Lopuhaä-Zwakenberg, M. (2025). Fault tree reliability analysis via squarefree polynomials: Mathematical and experimental analysis. *SN Computer Science 6*(8), 965.

Rauzy, A. (1993). New algorithms for fault trees analysis. *Reliability Engineering & System Safety 40*(3), 203–211.

Rauzy, A. (2008). Binary decision diagrams for reliability studies. *Handbook of performability engineering*, 381–396.

Reay, K. A. and J. D. Andrews (2002). A fault tree analysis strategy using binary decision diagrams. *Reliability engineering & system safety 78*(1), 45–56.

Sinnamon, R. M. and J. D. Andrews (1996). Fault tree analysis and binary decision diagrams. In *Proceedings of 1996 Annual Reliability and Maintainability Symposium*, pp. 215–222. IEEE.

Tolo, S. and J. Andrews (2022). An integrated modelling framework for complex systems safety analysis. *Quality and Reliability Engineering International 38*(8), 4330–4350.

Tolo, S. and J. Andrews (2023). Fault tree analysis including component dependencies. *IEEE Transactions on Reliability*, 1–9.

Tolo, S. and J. Andrews (2025). Modelling complexity in system safety: generalizing the d2t2 methodology. *Reliability Engineering & System Safety*, 112140.

Yan, D. and J. D. Andrews (2024). Fault tree culling in dynamic and dependent tree theory ($d^2t^2$) framework through approximation techniques. In *Advances in Reliability, Safety and Security*, Proceedings of ESREL 2024, Cham. Springer.