A new heuristics for the event ordering in binary decision diagram applied in fault tree analysis

Reni Banov¹, Zdenko Šimić² and Davor Grgić³

Abstract
Fault tree is a common approach in probabilistic risk assessment of complex engineering systems. Since their introduction, binary decision diagrams proved to be a valuable tool for complete quantification of hard fault tree models. As is known, the size of the binary decision diagram representation is mainly determined by the quality of the selected fault tree event ordering scheme. Finding the optimal event ordering for binary decision diagram representation is a computationally intractable problem, for which reason heuristic approaches are applied to find reasonable good ordering schemes. The existing method for finding optimal ordering schemes related to special types of fan-in 2 read-once formulas is employed in our research to develop a new heuristic for fault tree. Various fault tree simplification methods are used for the sake of reducing fault tree model discrepancy from fan-in 2 read-once formulas. The reduced fault tree is traversed in a depth-first manner, as for every gate, the best ordering scheme is chosen from selected sets of input permutations. The quality of the final event ordering scheme is compared to orderings produced with depth-first left most assignment approach. Our method proves to be a useful heuristic for finding good static event ordering, and it compares favourably to the known heuristic based on a depth-first left most assignment approach.

Keywords
Binary decision diagrams, event ordering, fault tree analysis, probabilistic risk assessment, reliability engineering

Introduction
The fault tree (FT) is frequently used to analyse safety and reliability requirements of complex engineering systems. At the beginning of the 1960s, FT was introduced by Bell Laboratories for the purposes of evaluating the Minuteman I launch control system. Afterwards it was further developed by numerous civil and military contributors in order to utilize computer-based probabilistic risk assessment (PRA) tools.¹ Usually, FT models are depicted graphically in form of a tree structure which represents a logical connection between possible events during the deductive analysis of the undesired top event.

Figure 1 presents a simple model where the top event (gate G1) shows logical connections between basic events (E1–E6) leading to G1 gate. This kind of logical connection can be written in form of logical functions. The graphical view below shows the logical function G1 = E6 ∨ (2, (E1 ∨ E2), E5, (E3 ∧ E4)) where G3 operation (tagged) represents ‘2-out-of-3’ voting OR logical expressions, meaning that it happens, if at least two of three inputs have occurred.

In general, FTs are used to investigate potential faults, modes and causes as well as to quantify their contribution to a system’s unreliability or to find importance factors for the components’ unavailability.² Traditionally, FT models are initially analysed qualitatively and then quantitatively. The qualitative analysis aims to find component failure lists which are essential and sufficient to cause the top event. These lists are often referred to as minimal cut sets and are widely

¹University of Applied Sciences, Zagreb, Croatia
²European Commission, Joint Research Centre (JRC), Petten, Netherlands
³Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia

Corresponding author:
Zdenko Šimić, European Commission, Joint Research Centre (JRC), P.O. Box 2, 1755ZZ Petten, Netherlands.
Email: Zdenko.SIMIC@ec.europa.eu
For clarity sake of the bottom-up approach, let us illustrate how to build BDD for the logical expression $(G4 \land G8)$ from the FT example (Figure 1), assuming that the variable ordering is defined as $E3 < E1 < E6 < E4 < E2 < E5$. Then

$$G4 = \text{ite}(E3, \text{ite}(E4, 1, 0), 0)$$

$$G8 = \text{ite}(E1, 1, \text{ite}(E2, 1, 0))$$

$$G4 \land G8 = \text{ite}(E3, \text{ite}(E4, 1, 0), \text{ite}(E1, 1, \text{ite}(E2, 1, 0)), 0)$$

Immediately after the Bryant seminal work, Friedman and Supowit\textsuperscript{9} created an algorithm of exponential complexity for finding the optimal variable ordering. Several other algorithms were created afterwards, yet they are applicable only for small sized problems, typically up to 50 variables. Computing an optimal variable ordering, and by this the event ordering for FT analysis, happens to be a non-deterministic polynomial-time (NP)-hard problem;\textsuperscript{10} however, the decision problem of improving variable ordering is NP-complete.\textsuperscript{11} Therefore, the hardness of the problem justifies the use of heuristic approaches for finding a good initial event ordering scheme for FTA\textsuperscript{12,13} or an event tree analysis (ETA).\textsuperscript{14}

The following section briefly explains an optimal ordering algorithm for a special kind of fan-in 2 read-once formulas. The remainder of the article is structured in the following order: section three explains the new heuristic approach along with FT simplification techniques, section four presents the used benchmarks and results achieved, providing an explanation of the advantages of the new approach. Finally, the last section brings the conclusion, encouraging further research.

Related work

Fan-in 2 read-once formulas on $n$ logical variables $\{x_1, \ldots, x_n\}$ are represented by a binary tree having exactly $n$ leaf nodes labelled with variables and internal nodes denoting one of the binary logical operations (AND, OR, XOR, $\ldots$) on two children nodes. For example, the following binary tree (Figure 2) represents the formula $(x_1 \land x_2) \lor (x_3 \land x_4) \land ((x_5 \lor x_6) \land x_7)$.

The optimal variable ordering scheme for fan-in 2 read-once formulas can be computed with an algorithm developed by Sauerhoff et al.\textsuperscript{15} Their algorithm is based on recursive equations for binary operations [and, or, xor]. At each gate, two numbers are calculated for optimal sizes of the BDDs with or without complemented edges. From these two numbers, the recursive equations define the rules for the computation of the optimal BDD size. As pointed out in Sauerhoff et al.,\textsuperscript{15} the optimal BDD size for a single variable requires one BDD node with complemented edges, or two without. Recursively, following the single variable
case upwards to the final formula, the authors presented a way to construct an optimal ordering by using only two sets of recursive equations, one set for \{and, or\} operations and another set for \(\text{xor}\) operation. In addition, they proved upper and lower bounds on an optimal BDD size for an arbitrary logical function on the restricted class of \(\text{fan-in 2 read-once}\) formulas. Since their algorithm is not directly applicable to FT models, it can be used in case where no overlapping between inputs to FT gate occurs, that is, in case where no formula sharing between gate inputs exists. Basically, it means that we can apply the respective algorithm with gates having only basic events as inputs or with gates having inputs from other gates without shared basic events. In addition, FT models considered in this article are models built with AND/OR gate operations, thereby enabling us to use a single set of recursive equations from their work.

**Method**

In this section, we are going to detail our newly developed heuristic approach to the variable ordering scheme throughout these subsections ‘FT simplification’ techniques arising from common logical equivalence rules; ‘New ordering heuristic’, which is the most important part of the new approach and ‘An outline of the algorithm’ with an optional BDD size reduction step.

**FT simplification**

Before the application of an ordering method, the FT structure has been simplified as much as possible. One of the approaches is to split complex FT models into simpler FTs by means of the functional decomposition with respect to a small subset of basic events.\(^{16}\) In this article, we perform a simpler approach based on logically equivalent formula rewriting rules\(^ {17}\) which may be applied iteratively as long as any changes are made in the FT, that is:

- The associativity rule – elimination of all equivalence gates by exchanging the references into gates they represent

**Figure 2.** Example of fan-in 2 read-once formula: \(((x_1 \land x_2) \lor (x_3 \land x_4)) \land ((x_5 \lor x_6) \land x_7)\).

**Figure 3.** The G3 gate operation expanded to equivalent formula.

- The K-of-N expansion rule – elimination of all K-of-N operators by rewriting them into equivalent AND/OR combinations according to the following rule of the gate expansion: \(k \cdot F_1, \ldots, F_n\)\(^{18}\) \(\left( (F_1 \land (k = 1, F_2, \ldots, F_n)) \lor (k, F_2, \ldots, F_n) \right)\) The K-of-N (voting OR) gate can be decomposed by more complex rules which we left for future implementations.
- Application of the idempotent law rules on the FT to eliminate the terms with multiple equal gate inputs: \(X \land X = X\), \((X \lor X = X)\).
- By applying the law of absorption rule to the AND/OR gates, the inputs are simplified by absorbing like terms: \((X \land Y) \lor X = X\), \((X \lor Y) \land X = X\).

From the given example, we show that the simple FT model has been modified by the application of K-of-N expansion rule (see Figure 3). The gate G3 operation is rewritten to the equivalent combination of the AND/OR operations (tagged) as follows

\[
G3 = (2, G8, E5, G4) = G31 \lor G32 \lor G33 \\
= (G8 \land E5) \lor (G8 \land G4) \lor (E5 \land G4)
\]

After the simplification steps, the FT has been traversed to find blocks of gates which behave like a single entity, that is, modularization.\(^ {19}\) Generally, two module finding algorithms are widely used to reduce the computational costs of BDD operations on the FT. The first one, derived by Kohda et al.,\(^ {20}\) is based on formula rewriting rules to detect and define modules on the FTs composed from the gates with logical AND/OR operation only. The second method developed by Dutuit and

\[\text{Banov et al. 399}\]
Rauzy uses a variation of the Tarjan algorithm for finding strongly connected graph components, thus being more universal since it does not depend on the gate order. Both algorithms perform equally well on the common FT models with only AND/OR gates, consequently we employ the latter method as it does not depend on the gate operation. For instance, the simple FT model presented here entails the following module top gates: G8, G4, G3 and G1.

New ordering heuristic

Based on the algorithm for fan-in 2 read-once formulas, we developed an ordering approach to sequentially combine the orders from the gate inputs. The idea is to utilize, as frequently as possible, the fan-in 2 read-once order combination technique on non-overlapping gate inputs. To be more specific, when orders from two gate inputs do not share events, the Sauerhoff, Wegener and Wercher algorithm is applied. However, if they do share events, they are combined according to the number of BDD nodes generated for each event in these orders. For implementation, each event in order is assigned the count of the BDD nodes without complemented edges. Our new order combining approach is defined by the following algorithm.

\[
\text{CombineOrders}(g, A, \pi)
\]

1. \( g \) – the gate for which we build an order
2. \( A \) – array of orders for the gates adjacent to \( g \)
3. \( \pi \) – the final order for \( g \)
4. \( n \) = \( \pi \)
5. For each order \( \pi_i \) from the array \( A \) do
   6. if \( (\pi_i \cap \pi) \neq \emptyset \) then
      7. \( \pi_i \leftarrow \emptyset \)
      8. \( n_{\pi_i} \leftarrow n \)
   9. For each event in orders \((\pi, \pi_i)\)
      10. Let \( e_1 \) be next event from \( \pi \) not present in \( \pi_n \)
      11. Let \( e_2 \) be next event from \( \pi_i \) not present in \( \pi_n \)
      12. if \( s_1 < s_2 \) then
         13. \( n_{\pi} \leftarrow n_{\pi} + n_{\pi_i} \)
         14. append \( e_1 \) to \( \pi \)
         15. append \( e_2 \) to \( \pi_i \)
         16. \( n_{\pi} \leftarrow n_{\pi} \)
      17. else
         18. append \( e_2 \) to \( \pi_n \)
         19. append \( e_1 \) to \( \pi_i \)
         20. \( n_{\pi} \leftarrow n_{\pi} \)
      21. end if
      22. end for
   23. end for
   24. \( \pi \leftarrow \pi_n \)
   25. else
      26. if \( s_1 < s_2 \) then
         27. \( n_{\pi} \leftarrow n_{\pi} + n_{\pi_i} \)
      28. append \( e_1 \) to \( \pi \)
      29. end if
   30. else
      31. \( n_{\pi} \leftarrow n_{\pi} + n_{\pi_i} \)
      32. append \( e_2 \) to \( \pi \)
      33. end if
   34. end if
   35. end for
   36. end if
   37. return \( \pi \)

Considering the above algorithm, the order for each gate is computed from the orders associated with the adjacent gates, which are considered from left to right (Line 5). In Line 6, we check the existence of shared events between orders; in case of their non-existence we proceed with the Sauerhoff algorithm (Lines 26–35). However, if we establish the existence of shared events the algorithm (Lines 7–24) alternatively proceeds with the implementation of our approach. For example, let us combine orders for gate G3 from intermediate gates after the simplification of the FT model. We assume that the orders

\[
\begin{align*}
\pi_1 &= \{(3,6) \{E5:(1,2), E1:(1,2), E2:(1,2)\}\} \\
\pi_2 &= \{(4,8) \{E3:(1,2), E4:(1,2), E1:(1,2), E2:(1,2)\}\} \\
\pi_3 &= \{(3,6) \{E5:(1,2), E3:(1,2), E4:(1,2)\}\}
\end{align*}
\]

are computed for gates G31, G32 and G33, respectively. On the first pass, through our approach the order \( \pi \) is initialized with \( \pi_1 \). Since each order entails a pair of \( n_{\pi_1,2} \) values followed by a list of basic events carrying their associated pair of \( n_{\pi_1,2} \) values, the loop iteration proceeds as follows:

\[
\begin{align*}
1. \quad &0: \pi = \{(3,6) \{E5:(1,2), E1:(1,2), E2:(1,2)\}\}, \quad \pi_1 = \{(3,6) \{E5:(1,2), E1:(1,2), E2:(1,2)\}\}, \\
2. \quad &1: \pi = \{(4,8) \{E3:(1,2), E4:(1,2), E1:(1,2), E2:(1,2)\}\}, \quad \pi_2 = \{(4,8) \{E3:(1,2), E4:(1,2), E1:(1,2), E2:(1,2)\}\}, \\
3. \quad &2: \pi = \{E5:(1,2), E1:(1,2), E2:(1,2), E3:(1,2), E4:(1,2)\}\}, \quad \pi_3 = \{(3,6) \{E5:(1,2), E3:(1,2), E4:(1,2)\}\}
\end{align*}
\]

are computed for gates G31, G32 and G33, respectively. On the first pass, through our approach the order \( \pi \) is initialized with \( \pi_1 \). Since each order entails a pair of \( n_{\pi_1,2} \) values followed by a list of basic events carrying their associated pair of \( n_{\pi_1,2} \) values, the loop iteration proceeds as follows:

\[
\begin{align*}
1. \quad &0: \pi = \{(3,6) \{E5:(1,2), E1:(1,2), E2:(1,2)\}\}, \quad \pi_1 = \{(3,6) \{E5:(1,2), E1:(1,2), E2:(1,2)\}\}, \\
2. \quad &1: \pi = \{(4,8) \{E3:(1,2), E4:(1,2), E1:(1,2), E2:(1,2)\}\}, \quad \pi_2 = \{(4,8) \{E3:(1,2), E4:(1,2), E1:(1,2), E2:(1,2)\}\}, \\
3. \quad &2: \pi = \{E5:(1,2), E1:(1,2), E2:(1,2), E3:(1,2), E4:(1,2)\}\}, \quad \pi_3 = \{(3,6) \{E5:(1,2), E3:(1,2), E4:(1,2)\}\}
\end{align*}
\]
12. \( e_1 = E1(2,4), e_2 = \{ \}, \quad \pi_n = \{(10,20) \{E5(2,4), E3(3,6), E4(3,6), E1(2,4)\}\} \)

13. \( e_1 = E2(2,4), e_2 = \{ \}, \quad \pi_n = \{(12,24) \{E5(2,4), E3(3,6), E4(3,6), E1(2,4), E2(2,4)\}\} \)

14. \( \pi = \pi_n = \{(12,24) \{E5(2,4), E3(3,6), E4(3,6), E1(2,4), E2(2,4)\}\} \)

By this a new order from gate inputs is computed, which herein becomes the actual order for gate G3.

**An outline of the algorithm**

As it is known, in the conventional approach to FTA the model is traversed in a depth-first manner, whereby actions are performed on each encountered gate. In our approach we are going to apply the FindBestOrder procedure to compute the proposed order for the observed gate. For instance, the order for G1 gate from the FT example is built by means of the FindBestOrder procedure on the following gates: G8, G4, G31, G32, G33, G3 and G1. By applying the FindBestOrder procedure we intend to generate different permutations of adjacent gate inputs and employ the new heuristic to combine the orders from them. Among all generated orderings for the specific gate, the order with the smallest BDD in size will be selected.

As presented in the above algorithm, the initialization is performed at the beginning of the procedure (Lines 4–6). The result of this initialization (denoted as \( \pi_F \)) is achieved by employing our heuristic to combine the orders from gate inputs which are module top gates. Following that, we set the two limits for the underlying BDD engine, that is, the time limit and the maximum number of nodes allowed. The purpose of this is to reject orders which are inefficient in terms of time or nodes to be finished. The remaining adjacent gates are permuted and, by applying our heuristic, combined with \( \pi_F \) to create a complete order for the gate; subsequently, it is checked and best in size selected (Lines 8–19). The permutation loop is limited to 120 repetitions, which makes good trade-offs between the quality of the generated order and time to find it.

For further understanding of this procedure, let us study an example with G3 inputs \( \{G31, G32, G33\} \), which are entailed in the array of orders A from adjacent gates. Within the loop we use all possible permutations of this array

\[
\{G31, G32, G33\}, \{G31, G33, G32\}, \{G32, G31, G33\},
\{G32, G33, G31\}, \{G33, G31, G32\}, \{G33, G32, G31\}
\]

as the input to our heuristic. With the intention to build the order for top gate G1 we must combine the order for gate G3 with event E6. In the earlier presented example for CombineOrders heuristic we created the order for G3 gate, which is

\[
\pi_{G3} = \{(12,24) \{E5 : (2,4), E3 : (3,6), E4(3,6), E1 : (2,4), E2 : (2,4)\}\}
\]

Both, E6 as the directly connected adjacent event and G3 as module top gate are combined with our heuristic (Lines 4–6) to the final order for G1 gate

\[
\pi = \{(13,26) \{E6 : (1,2), E5 : (2,4), E3 : (3,6), E4(3,6), E1 : (2,4), E2 : (2,4)\}\}
\]

An interesting fact arising from the upper order of our example is that, actually we expected 13 BDD nodes for the FT model, however, once it was built it eventually resulted in 10 nodes.

The generally used method for the reduction of BDD size is the dynamic reordering following the construction step, which in this case is optional (Line 20). There are various dynamic reordering schemes based on the swap of two adjacent variables in an algorithm developed by Rudell.\(^{22}\) Commonly used methods are group sifting\(^{23}\) and minimization by exploiting of strong symmetries between variables.\(^{24,25}\) For this paper we selected the group sifting method as an optional BDD size reduction step (Line 20). For instance, the generated order from the algorithm is reduced from 10 to 9 BDD nodes for the FT model in our example, which is an optimal number of nodes for such a small example. As the results will show, the optional dynamic reordering step proves to be useful, and more importantly, being the key step for a successful construction of BDDs for large FT models.

---

**FindBestOrder( g, A, \( \pi \))**

1. \( g \) – the gate for which we build an order
2. \( A \) – array of orders for the gates adjacent to \( g \)
3. \( F \) – array of orders which are fixed for all generated permutations of \( A \)
4. < Move basic events from \( A \) to \( F \) which are directly adjacent to \( g \) >
5. < Move orders from \( A \) to \( F \) which are module tops >
6. \( \pi \leftarrow \text{CombineOrders}(\text{g}, \text{F}, \emptyset) \)
7. < Initialize calculations >
8. For each new permutation \( \pi_n \) from \( A \) do
9. \( \pi_n \leftarrow \text{CombineOrders}(\text{g}, \pi_n, \pi_F) \)
10. < Build BDD of \( g \) with order \( \pi_n \) >
11. \( \text{size} \leftarrow \text{number of BDD nodes for gate} \ g \ \text{with order} \ \pi_n \)
12. if \( \text{size} < \text{best_so_far_size} \) then
13. \( \pi \leftarrow \pi_n \)
14. \( \text{best_so_far_size} \leftarrow \text{size} \)
15. end if
16. if number of generated permutations greater than some limit then
17. Exit For
18. end if
19. end for
20. < Apply optional dynamic reordering step to the order \( \pi \) >
21. return \( \pi \)
Results and discussion

This section presents the test results obtained from applying the new ordering heuristic to selected sets of FT models from the literature\textsuperscript{26,27} and to a set of own large FTs from industry applications. The proposed ordering heuristic is compared to the best results obtained from depth-first left most (DFLM). The Colorado University Decision Diagram (CUDD) BDD package is used to implement DFLM as well as the new algorithm presented in this article. Since it is clear that there is no benefit from selecting gate inputs in any order,\textsuperscript{28} we decided to perform tests with DFLM and our heuristic for 100 times with a randomized ordering of inputs at each gate. For a better assessment of the results gained with the new method, the following statistics for top gate BDD size of each FT model were collected: minimum number of nodes, average number of nodes and maximum number of nodes out of 100 runs.

Comparison on small-to-medium FT models

In the first set of tests from the literature, we used 41 small-to-medium sized FTs from the Aralia benchmark set,\textsuperscript{26,27} that is, tests with less than 1000 events and gates aggregative. These FTs originate from different sources and have different characteristics in terms of size, structure, and number of internal connections among the gates and basic events. The results of the tests are summarized in Table 1, both for the DFLM

| Fault tree model | DFLM heuristic BDD size | New heuristic BDD size |
|------------------|--------------------------|------------------------|
|                  | Name            | Events | Gates | Min. | Avg. | Max. | Min. | Avg. | Max. |
| baobab1          | 61              | 84     | 1615  | 5194 | 16,162 | 2025 | 2207 | 2439 |
| baobab2          | 32              | 40     | 197   | 346  | 494    | 226  | 259  | 290  |
| baobab3          | 80              | 107    | 2992  | 11,461 | 26,509 | 3582 | 3995 | 4341 |
| cays601          | 186             | 201    | 11,415 | 251,753 | 1.5M | 5400 | 8185 | 10,042 |
| chinese          | 25              | 36     | 73    | 105  | 143    | 66   | 68   | 71   |
| dss2001          | 122             | 82     | 569   | 775  | 898    | 191  | 230  | 273  |
| dss2002          | 49              | 36     | 108   | 236  | 347    | 50   | 79   | 108  |
| dss2003          | 51              | 30     | 60    | 111  | 179    | 55   | 59   | 62   |
| dss2004          | 53              | 30     | 61    | 89   | 122    | 53   | 53   | 53   |
| dss2005          | 51              | 20     | 52    | 52   | 52     | 52   | 52   | 52   |
| dss2006          | 121             | 112    | 449   | 1388 | 2340   | 474  | 546  | 636  |
| dss2007          | 276             | 324    | 2934  | 13,375 | 47,178 | 3143 | 5395 | 13,229 |
| dss2008          | 103             | 145    | 1186  | 2131 | 5240   | 1270 | 1357 | 1516 |
| dss2009          | 109             | 73     | 110   | 137  | 191    | 111  | 156  | 200  |
| dss601           | 122             | 288    | 1277  | 2343 | 4140   | 1109 | 1444 | 1843 |
| edf2001          | 183             | 132    | 1706  | 2094 | 2910   | 1302 | 1391 | 1446 |
| edf2002          | 458             | 435    | 471   | 689  | 1841   | 440  | 488  | 577  |
| edf2003          | 362             | 475    | 8381  | 20,892 | 74,845 | 8338 | 38,372 | 55,828 |
| edf2004          | 323             | 375    | 6864  | 30,958 | 89,934 | 22,544 | 25,458 | 54,402 |
| edf2005          | 165             | 142    | 448   | 1011 | 2037   | 471  | 505  | 554  |
| edf2006          | 240             | 362    | 733   | 4095 | 17,880 | 907  | 1477 | 1849 |
| edfpa14b         | 311             | 290    | 6616  | 21,735 | 63,683 | 7890 | 22,916 | 52,043 |
| edfpa14d         | 311             | 173    | 6829  | 21,805 | 60,925 | 9563 | 17,506 | 38,646 |
| edfpa14p         | 124             | 101    | 2855  | 10,401 | 35,887 | 6579 | 15,526 | 39,259 |
| edfpa14q         | 311             | 194    | 7638  | 23,304 | 79,589 | 4990 | 7108 | 12,336 |
| edfpa14r         | 106             | 132    | 2817  | 9982  | 38,363 | 2714 | 13,951 | 51,827 |
| edfpa15b         | 283             | 249    | 2400  | 7542  | 18,808 | 2322 | 3700 | 11,820 |
| edfpa15d         | 283             | 138    | 2356  | 7910  | 20,817 | 4111 | 6170 | 9567 |
| edfpa15p         | 100             | 80     | 914   | 4326  | 14,906 | 1703 | 2522 | 7715 |
| edfpa15q         | 289             | 151    | 2175  | 7677  | 23,879 | 1698 | 2013 | 2333 |
| edfpa15r         | 88              | 110    | 861   | 3543  | 9990   | 1005 | 1521 | 3227 |
| elf9601          | 145             | 242    | 962   | 3114  | 7620   | 638  | 652  | 665  |
| ftr10            | 175             | 94     | 289   | 1144  | 1928   | 181  | 220  | 297  |
| isp9601          | 143             | 104    | 383   | 853   | 2193   | 238  | 283  | 346  |
| isp9602          | 116             | 122    | 529   | 1302  | 2313   | 444  | 452  | 459  |
| isp9603          | 91              | 95     | 440   | 1464  | 3203   | 443  | 481  | 522  |
| isp9604          | 215             | 132    | 701   | 1103  | 1930   | 487  | 518  | 525  |
| isp9605          | 32              | 40     | 185   | 439   | 748    | 244  | 244  | 244  |
| isp9606          | 89              | 41     | 142   | 283   | 550    | 108  | 137  | 184  |
| isp9607          | 74              | 65     | 274   | 469   | 643    | 122  | 130  | 140  |
| jbd-001          | 533             | 315    | 5248  | 321,132 | 321,132 | 8569 | 12,191 | 24,280 |
| Average          |                 |        |       | 1.1   |        | 2.8  |        | 7.6   |

DFLM: depth-first left most; BDD: binary decision diagram.

Table 1. Results for the comparison of DFLM and new heuristic on small to medium FT models from the Aralia benchmark set.
and for the herein presented new approach. It is important to mention that all runs (for DFLM and the new heuristic) give variable orderings with which the BDD from the FT model was constructed within the given limits (up to 100M nodes) on the CUDD package. The results are also presented graphically in Figure 4.

The comparison shows that the best results for the new method are nearly the same as the best results achieved with the DFLM method (column Min. in Table 1). Comparing average ratios of BDD results, the new heuristic performs better than the DFLM to an average of almost three times (precisely 2.8). What is more important, in most cases the new heuristic outperforms the average DFLM results by several times, even 26 times in one case (cea9601). In addition to that, the new method appears rather stable in producing good orderings for the analysed FTs, since the average ratio between the maximum and minimum size is less than three, while for the DFLM approach it is more than 11 (more than eight if cea9601 is thrown). This proves equally the stability of the approach as well as the quality of the orders obtained by our heuristic approach.

In addition to that, a single run was made with the new heuristic to check the correlation between the estimated size and built size without sifting (Table 3 in Appendix 1). On average, the estimated size is approximately 4.5 times bigger than the BDD built size and it is achieved in a reasonable time. The longest running time for the heuristic was approximately 5 min on the edf9202 model, which is still acceptable. It is worthwhile mentioning that in one test (das9207), the estimated size from our heuristic proved to be extremely overestimated, approximately 100 times more than the built size. Apart from that extreme result, the estimated size from our heuristic is on average two times bigger than the built size. After the optional reordering method was applied on built BDDs, the size further lowered on average by a factor of 1.85, that is, approximately 46% less BDD nodes are needed to represent a selected benchmark set.

**Comparison on large FT models**

In addition to tests performed on the Aralia benchmark set, we selected some of own larger FT models from the nuclear industry, that is, FT models in which the sum of gates and events exceed 1000. The largest selected model (i.e. cored2) reaches the limits of the CUDD engine, while others happen to be too large for a plain DFLM heuristic. Therefore, we decide to implement the DFLM

![Figure 4. Graphical view of results for the comparison of DFLM (left bar/black) and the new heuristic (right bar/orange) on small-to-medium FT models from the Aralia benchmark set.](image)

**Table 2.** Results for the comparison of DFLM and new heuristic for large FT models.

| Fault tree model | DFLM heuristic BDD size | New heuristic BDD size |
|------------------|-------------------------|------------------------|
|                  | Min. | Avg. | Max. | Min. | Avg. | Max. |
| acp              | 674  | 409  |      | 15,705 | 31,972 | 3047  |
| chrgr            | 695  | 438  |      | 348,155 | 1.2M | 25,305  |
| cored1           | 1279 | 1319 |      | n/a  | n/a  | 1.2M |
| cored2           | 1633 | 1377 |      | n/a  | n/a  | 23M |
| dcp              | 706  | 447  |      | 265,403 | 1.1M | 26,660  |
| efw              | 957  | 692  |      | 7.5M | 350,408 | 2M |
| hpsi             | 940  | 674  |      | 480,635 | 480,635 | 21,369  |
| lpsi             | 760  | 525  |      | 18,962 | 44,505 | 9516  |
| sw               | 720  | 444  |      | 379,953 | 379,953 | 19,421  |
| Average size ratio DFLM/new | 1.9 | 5.4 | 13.0 |

DFLM: depth-first left most; BDD: binary decision diagram.
heuristic with an automatic dynamic reordering (the group sifting method) during the BDD construction phase which is a common approach in constructing BDDs for large models. For such an extensive test, we employ the optional dynamic reordering step in FindBestOrder procedure on gate orders expecting a number of BDD nodes within the range from 1M up to 10M nodes. The tests were run for 10 times with an increased limit on the CUDD package to 1000M nodes.

The results are summarized in Table 2 and graphically presented in Figure 5. As shown in Table 2, the DFLM heuristic fails to produce the ordering within the limits on ‘cored1’ and ‘cored2’ FTs, while in other tests the DFLM heuristic, compared to our approach, performs consistently worse. In terms of the stability for the new method, the model complexity has moderate impact, since the average ratio between maximum and minimum results is less than three, similar to small-to-medium models. In the last row, it is possible to compare averaging ratios, excluding the ‘cored1’ and ‘cored2’ tests. Here, even for the best results, the new approach is consistently, nearly two times, better compared to the DFLM with automatic dynamic orderings applied during the construction phase. The average results prove that our approach performs far better yielding a ratio over five, while in one case the ratio surges convincingly higher (to 11.7 for ‘chrgr’).

**Discussion**

Considering small-to-medium sized models, both heuristics produce orderings with which the BDD representation for every FT model was built. The results on tests with small-to-medium sized models show that on the average, our heuristic produces better event ordering than the DFLM heuristic. Moreover, it can be noticed that the optional dynamic reordering proves to be helpful for both approaches. However, when applied in our heuristic it gives better results than an ordinary application of reordering at the end of the BDD creation step. The results are somewhat better for the DFLM heuristic on minimum sizes in a respectful number of cases (18 from 41). This is assuming the result of a random selection of gate inputs in both approaches.

Finally, in the analysis of larger FT models (Table 2) the new ordering heuristic is able to produce an order with which we build the BDD representation, even in the case of the largest considered models, where DFLM heuristic failed. This indicates that our approach does not only bring better results but also has a higher capability of beneficial order finding. Additional tests on large FT models prove that dynamic reordering during the build phase is not in all cases helpful for DFLM heuristic, yet it is useful in the new heuristic. It is far better to perform dynamic reordering after the BDD has been constructed and to refrain from automatic reordering during the build phase. The cause may be that automatic reordering within the BDD build phase simply does not sufficiently dispose over information on the event dependency as they appear to be available after the BDD has been constructed.

**Conclusion**

The new heuristic emerges from the known approach for fan-in 2 read-once formulas which has been further extended to a mixed case. Combining the heuristics with the simple exhaustive search, the new complete ordering algorithm is developed and verified on test models with different structural characteristics. The results from tests performed with the new algorithm show a clear improvement over the widely used DFLM heuristic.

Also, some important positive facts as well as some important issues have been discovered. The DFLM heuristic is very sensitive to the order of gate inputs. Our ordering algorithm based on the fan-in 2 read-once formulas algorithm appears useful even when applied to the BDD construction phase on large FT models with more than 1000 events and gates, the automatic reordering is of no help.

The presented approach proves to be promising for the application on industry FT models and indicates that further research may show potential in direction of: simplification of FT models to be more like fan-in k read-once formulas; application of search strategies other than exhaustive search at the gate level when orders from adjacent gate inputs are combined;
investigation of different combining techniques (weight-based, greedy, AI search, ...) on overlapping orders from adjacent gate inputs; application of dynamic reordering at different points during the order construction.

Apart from the presentation of the innovative heuristic, this article also reveals inspiring ideas for further research and development of an improved event ordering heuristic for the BDD-based analysis of FT models.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship and/or publication of this article.

Funding
The author(s) received no financial support for the research, authorship and/or publication of this article.

ORCID iDs
Reni Banov https://orcid.org/0000-0002-0355-4880
Zdenko Šimic https://orcid.org/0000-0001-7514-8227

References
1. Nusbaumer O and Rauzy AB. Fault tree linking versus event tree linking approaches: a reasoned comparison. P I Mech Eng O-J Ris 2013; 227(3): 315–326.
2. Dutuit Y and Rauzy AB. Importance factors of coherent systems: a review. P I Mech Eng O-J Ris 2014; 228(3): 313–323.
3. Lee CY. Representation of switching circuits by binary decision programs. Bell Syst Tech J 1959; 38: 985–999.
4. Akers SB. Binary Decision Diagrams. IEEE T Compt 1978; 27(6): 509–516.
5. Adam A. Truth functions and the problem of their realization by two-terminal graphs. Budapest: Akademia Kiado, 1969.
6. Bryant RE. Graph-based algorithms for Boolean function manipulation. IEEE T Compt 1986; C-35(8): 677–691.
7. Wegener I. BDDs-design, analysis, complexity, and applications. Discrete Appl Math 2004; 138(1–2): 229–251.
8. Rauzy AB. New algorithms for fault tree analysis. Reliab Eng Syst Safe 1993; 40(3): 203–211.
9. Friedman SJ and Supowit KJ. Finding the optimal variable ordering for Binary Decision Diagrams. IEEE T Compt 1990; C-39(5): 710–713.
10. Bolling B. On the complexity of some ordering problems. In: Csuhaj-Varjú E, Dietzfelbinger M and Esik Z (eds) Mathematical foundations of computer science 2014 (Lecture notes in computer science), vol. 8635. Berlin: Heidelberg: Springer, 2014, pp.118–129.
11. Bolling B and Wegener I. Improving the variable ordering of OBDDs is NP-complete. IEEE T Compt 1996; C-45(9): 993–1002.
12. Yuchang M, Farong Z, Huawen L, et al. Depth-first event ordering in BDD-based fault tree analysis. Comput Inform 2013; 31(6): 1401–1416.
13. Yuchang M, Farong Z, Huawen L, et al. Efficient ordering heuristics in Binary Decision Diagram-based fault tree analysis. Qual Reliab Eng Int 2013; 29(3): 307–315.
14. Ibanez-Llano C, Melendez E and Nieto F. Variable ordering schemes to apply to the Binary Decision Diagram methodology for event tree sequences assessment. P I Mech Eng O-J Ris 2008; 222(1): 7–16.
15. Sauerhoff M, Wegener I and Werchner R. Optimal ordered Binary Decision Diagrams for read-once formulas. Discrete Appl Math 2000; 103(1–3): 237–258.
16. Matuzas V and Contini S. On the efficiency of functional decomposition in fault tree analysis. P I Mech Eng O-J Ris 2012; 226(6): 635–645.
17. Mordechai B. Mathematical logic for computer science. 2nd ed. London: Springer-Verlag, 2001.
18. Wang J, Chen S, Wang F, et al. A new decomposition algorithm for complex voting gates processing in qualitative fault tree analysis. P I Mech Eng O-J Ris 2015; 229(6): 568–575.
19. Remenyte R and Andrews JD. Qualitative analysis of complex modularized fault trees using Binary Decision Diagrams. P I Mech Eng O-J Ris 2006; 220(1): 45–53.
20. Kohda T, Henley EJ and Inoue K. Finding modules in fault trees. IEEE T Reliab 1989; 38(1): 165–176.
21. Dutuit Y and Rauzy AB. A linear-time algorithm to find modules of fault trees. IEEE T Reliab 1996; 45(3): 422–425.
22. Rudell R. Dynamic variable ordering for ordered Binary Decision Diagrams. In: Proceedings of the 1993 IEEE/ACM international conference on computer aided design, San Jose, CA, 7–11 November 1993, pp.42–47. New York: IEEE.
23. Panda S and Someni F. Who are the variables in your neighborhood. In: Proceedings of the 1995 IEEE/ACM international conference on computer aided design (ICCAD1995), San Jose, CA, 5–9 November 1995, pp.74–77. New York: IEEE.
24. Panda S, Someni F and Plessier BF. Symmetry detection and dynamic variable ordering of decision diagrams. In: Proceedings of the 1994 IEEE/ACM international conference on computer aided design, San Jose, CA, 6–10 November 1994, pp.628–631. New York: IEEE.
25. Scholl C, Melchior S, Hotz G, et al. Minimizing ROBDD sizes of incompletely specified Boolean functions by exploiting strong symmetries. In: Proceedings European design and test conference, Paris, 17–20 March 1997, pp.229–234. New York: IEEE.
26. Rauzy AB. Some disturbing facts about depth-first leftmost variable ordering heuristics for Binary Decision Diagrams. P I Mech Eng O-J Ris 2008; 222(4): 573–582.
27. Groupe Aralia. Computation of prime implicants of a fault tree within Aralia. In: Proceedings of the European safety and reliability association conference (ESREL 1995), Bournemouth, 26–28 June 1995, pp.190–202. London: ESRA.
28. Yuchang M, Liudong X, Farong Z, et al. Choosing a heuristic and root node for edge ordering in BDD-based network reliability analysis. Reliab Eng Syst Safe 2014; 131: 83–93.
Appendix 1

Table 3 contains the statistics from a single run of the new heuristic on a complete set of 41 FT, small to medium sized, from the Aralia benchmark set. The statistics were collected without optional dynamic variable ordering in the procedure FindBestOrder. The first column shows the estimated number of BDD nodes resulting from the order produced with our heuristic, while the next column shows the number of BDD nodes used when it was built with CUDD engine. The third column brings the number of nodes after the group sifting method was applied on built BDD. The fourth column presents the CUDD engine report showing the peak number of BDD nodes used during the built process with that order. Eventually, the last column shows the time needed by the new heuristic to find that order.

Table 3. Statistics on estimated and built performance of the new heuristic.

| Fault tree model | New heuristic execution statistics |  |  |  |
|------------------|------------------------------------|----|----|----|
|                  | Estimated BDD size | Number of nodes as built | BDD size after sifting | Peak BDD nodes | Time to find order (s) |
| baobab1          | 3489                 | 1828                        | 1398                      | 8176             | 10.1                     |
| baobab2          | 523                  | 494                         | 114                       | 7154             | 5.4                      |
| baobab3          | 5676                 | 4215                        | 3285                      | 15,330           | 5.1                      |
| cea9601          | 6639                 | 6070                        | 5538                      | 54,166           | 31.3                     |
| chinese          | 85                   | 63                          | 45                        | 1022             | 0.6                      |
| das9201          | 642                  | 525                         | 179                       | 4088             | 9.8                      |
| das9202          | 144                  | 98                          | 62                        | 1022             | 3.4                      |
| das9203          | 115                  | 67                          | 52                        | 1022             | 1.6                      |
| das9204          | 120                  | 69                          | 53                        | 1022             | 3.1                      |
| das9205          | 101                  | 52                          | 52                        | 1022             | 1.1                      |
| das9206          | 1427                 | 632                         | 468                       | 4088             | 5.0                      |
| das9207          | 1,215,274            | 12,119                      | 2608                      | 105,266          | 35.9                     |
| das9208          | 1713                 | 1312                        | 1015                      | 7154             | 19.8                     |
| das9209          | 258                  | 151                         | 110                       | 2044             | 5.1                      |
| das9601          | 2747                 | 2159                        | 1641                      | 143,080          | 41.1                     |
| edf9201          | 1928                 | 1706                        | 1528                      | 12,264           | 2.1                      |
| edf9202          | 1317                 | 1113                        | 442                       | 119,574          | 309.8                    |
| edf9203          | 11,087               | 7135                        | 5941                      | 303,534          | 108.0                    |
| edf9204          | 52,050               | 26,922                      | 24,301                    | 123,662          | 70.8                     |
| edf9205          | 1342                 | 632                         | 376                       | 5110             | 24.8                     |
| edf9206          | 1879                 | 1583                        | 1030                      | 28,616           | 199.5                    |
| edf9207          | 13,079               | 8877                        | 6566                      | 51,122           | 30.2                     |
| edfpa14b         | 51,290               | 8617                        | 4411                      | 57,232           | 64.8                     |
| edfpa14o         | 23,218               | 6164                        | 2019                      | 44,968           | 10.7                     |
| edfpa14p         | 6703                 | 5742                        | 3929                      | 52,122           | 144.9                    |
| edfpa14q         | 5919                 | 5616                        | 5258                      | 20,440           | 16.9                     |
| edfpa15b         | 5272                 | 3745                        | 3053                      | 21,462           | 22.6                     |
| edfpa15o         | 9944                 | 3955                        | 3127                      | 33,726           | 37.1                     |
| edfpa15q         | 22,693               | 2186                        | 1073                      | 8176             | 6.3                      |
| edfpa15r         | 5407                 | 2947                        | 2022                      | 17,374           | 64.5                     |
| elf9601          | 1827                 | 1206                        | 666                       | 8176             | 9.4                      |
| elt9601          | 982                  | 843                         | 840                       | 45,990           | 40.1                     |
| ftr10            | 677                  | 504                         | 184                       | 6132             | 1                        |
| isp9601          | 873                  | 732                         | 366                       | 5110             | 5.6                      |
| isp9602          | 731                  | 617                         | 606                       | 4088             | 8.3                      |
| isp9603          | 2983                 | 654                         | 426                       | 3066             | 4.7                      |
| isp9604          | 1090                 | 749                         | 519                       | 6132             | 9.6                      |
| isp9605          | 575                  | 546                         | 123                       | 8176             | 5.2                      |
| isp9606          | 357                  | 270                         | 108                       | 2044             | 3.0                      |
| isp9607          | 474                  | 405                         | 124                       | 2044             | 3.7                      |
| jbd-001          | 87,261               | 14,469                      | 6055                      | 118,552          | 7.4                      |

Average ratios: Est./built 4.45, Built/sifted 1.85

BDD: binary decision diagram.