Comparison of variable ordering heuristics / algorithms for binary decision diagrams

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Citation: BARLETT, L.M. and ANDREWS, J.D., 1999. Comparison of variable ordering heuristics / algorithms for binary decision diagrams. IN: Proceedings of SARS99 Annual Safety and Reliability Society Conference, Manchester

Additional Information:

- This is a conference paper.

Metadata Record: [https://dspace.lboro.ac.uk/2134/3829](https://dspace.lboro.ac.uk/2134/3829)

Publisher: © L.M. Bartlett and J.D. Andrews

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
COMPARISON OF VARIABLE ORDERING HEURISTICS/ALGORITHMS
FOR BINARY DECISION DIAGRAMS

L.M.Bartlett and J.D.Andrews
Department of Mathematical Sciences
Loughborough University
Loughborough
Leicestershire
LE11 3TU

ABSTRACT
Fault tree analysis is a commonly used technique to assess the systems reliability performance in terms of its components reliability characteristics. More recently, the Binary Decision Diagram (BDD) methodology has been introduced which significantly aids the analysis of the fault tree diagram. The approach has been shown to improve both the efficiency of determining the minimal cut sets of the fault tree, and also the accuracy of the calculation procedure used to quantify the top event parameters.

To utilise the technique the fault tree structure needs to be converted into the BDD format. Converting the fault tree is relatively straightforward but requires the basic events of the tree to be placed in an ordering. The ordering of the basic events is critical to the resulting size of the BDD, and ultimately affects the performance and benefits of this technique.

Numerous studies have tackled this variable ordering problem and a number of heuristic approaches have been developed to produce an optimal ordering permutation for a specific tree. These heuristic approaches do not always yield a minimal BDD structure for all trees, some approaches generate orderings that are better for some trees but worse for others. The most recent research to find an approach to produce an optimal ordering for a range of trees has looked at pattern recognition approaches, such as genetic algorithm based classifier systems.

This paper reviews the heuristic approaches that have been established and examines the pattern recognition techniques that have been applied more recently. Another potential new algorithm for ordering using the structural importance of the components is proposed.
INTRODUCTION

Over the past five years an alternative technique, to Kinetic Tree Theory (Vesely [1]), known as the Binary Decision Diagram (BDD) method has been developed [2-6] to analyse the fault tree. This method has proved to be more accurate and efficient than the conventional approach. In calculating the system or top event parameters it does not need to first evaluate all the minimal cut sets, nor does it require the use of approximations, the exact calculations can be performed.

To use the BDD methodology the fault tree representing the system failure mode must first be converted to a BDD. To accomplish this the basic events in the fault tree are placed in an order. A good ordering of the basic events can result in a very efficient analysis, a poor ordering can lead to problems.

Several research papers have been published which investigate different ordering strategies and heuristics [6-13]. From the research to date a number of heuristics have been developed which are effective for specific fault tree structures, but a general heuristic that produces a minimal BDD for all fault trees is not available. The latest research looks at rule based approaches [14] to identify an ordering scheme which yields an efficient ordering of the fault tree variables. Lack of this efficient ordering for any tree structure is probably the reason that only one commercially available code [15] has been produced which is based on this method.

BINARY DECISION DIAGRAMS

A BDD is a directed acyclic graph, as shown in figure 1. All paths through the BDD start at the root vertex and terminate in one of two states, either a 1 state which corresponds to a system failure, or a 0 state which corresponds to a system success. A BDD is composed of terminal and non-terminal vertices, which are connected by branches. Non-terminal vertices correspond to the basic events of the fault tree.

All the left branches leaving a vertex are the 1 branches (component failure occurs) and all the right branches the 0 branches (component functional). Every path starts from the root vertex, and proceeds down through the diagram to the terminal vertices.
Only the vertices that lie on a 1 branch on the way to a terminal 1 vertex are included in the path. All the paths terminating in a 1 state give the cut sets of the fault tree. For example, the cut sets of figure 1 are:

1) $X_1X_2X_3$  
2) $X_1X_4$.

The method to convert a fault tree to its equivalent BDD is described in many publications and the reader is referred to them for details (refs. [3],[16]).

**FIGURE 1**
A Binary Decision Diagram

![Binary Decision Diagram](image)

**The Variable Ordering Dilemma**

In constructing the BDD, the ordering of the basic events is crucial to the size of the resulting diagram. Using an inefficient ordering scheme will produce a non-minimal BDD structure. Alternative ordering schemes will produce BDD’s of different sizes, the smaller the BDD the more optimal the diagram. To illustrate this fact, consider the simple fault tree shown in figure 2. The tree has four basic events, where $X_2$ is repeated.

If the basic event ordering permutation of $X_1<X_2<X_3<X_4$ is taken, the resulting BDD is shown in figure 3. This structure consists of only four nodes, it is a minimal structure and hence produces only minimal cut sets.
However, if the alternative ordering permutation of $X_4<X_3<X_2<X_1$ is taken the resulting BDD (shown in figure 4) consists of seven nodes, it is non-minimal and yields non-minimal cut sets. For larger fault tree structures the efficiency of the resulting BDD is more critical, and in the worst case of using a poor ordering permutation, the diagram may be unsolvable.
HEURISTIC APPROACHES IN THE LITERATURE

A number of different heuristics have been published in the literature, and the more influential ones are highlighted in this paper. The most common heuristic for ordering is produced by listing the variables in a top-down, left-right basis from the original fault tree structure. However, BDD's produced using this simple ordering of the variables are frequently inefficient since they produce a large number of non-minimal cut sets. The non-minimum BDD must undergo the minimising procedure to obtain the minimal cut sets, which can cause an undesirable increase in computer time. Also using the non minimised BDD is inefficient when calculating the top event probability and frequency of occurrence. It is therefore beneficial to achieve an ordering which is optimal in terms of the resulting size of the BDD.

An alternative ordering, to the common top-down approach, is presented by Sinnamon and Andrews [6]. The scheme focuses on those basic events that are repeated in the fault tree structure. It is the repeated events which cause the problem of non-minimal cut sets, and by considering these events first simplifies the resulting BDD.
structure and can therefore make it more optimal. The alternative ordering scheme still uses the top-down algorithm, however, as each gate is considered the basic events which are inputs to the gate are taken in order with those which occur most frequently placed first in the ordering list. When gate inputs are encountered which are already entered in the ordering list due to their occurrence at a higher level in the tree then they are ignored and the remaining events ordered. The results suggest that the new ordering appears to produce more optimal BDD's compared to other orderings.

Later research by the same authors [7] compare the results of some additional heuristics which again consider the repeated events. Three different ordering heuristics were considered in this paper to investigate the effects that different schemes produce. These orderings were:

1. Top-down ordering;
2. Top-down ordering with each subtree ordered first (depth-first);
3. As (2) and gates with only basic event inputs are ordered first (priority depth-first).

Within each of the schemes there is an option to list repeated events first. This produces six ordering permutations. The top-down, left-right approach is the most common, and is produced by listing the variables in a top-down, left-right manner from the original fault tree structure. The depth first approach involves breaking the whole tree structure into smaller trees (subtrees) and looking at the optimal ordering of these subtrees. The depth first ordering scheme gives each subtree a top-down, left-right ordering, working from the first gate inputs of the top event. The priority depth first approach takes the depth first approach one step further and considers subtrees with only basic event inputs first. The three additional ordering heuristics give priority to the repeated events. If the gate has more than one repeated event as an input then the most repeated event is placed first, if they occur the same number of times then the events are taken in gate list order to break the tie.

The results show that there are vast differences in the number of computations required to construct the BDD when different orderings are used for the basic events. Hence, great savings can be made in terms of computation time and memory requirements when an efficient ordering of the basic events can be established.
However, it is clear from the examples that each tree has an individual variable ordering that will optimise its size. There doesn't appear to be a general ordering scheme that will be 'best' for all trees.

Bouissou et al., [9] compared and evaluated six different variable ordering heuristics. Some of these heuristics were already discussed in the literature and in addition the authors proposed two new ones. The research showed that the results of the performance of each heuristic were related to the task to be completed. If the main objective of a study was to obtain a first BDD as soon as possible, then using a heuristic which allocated weights to each variable [10] or that of considering the number of connectives of each gate, should be used. However, if on the other hand, the objective was to obtain a BDD as small as possible, then using a depth-first traversal or heuristic related to the number of fanouts [11] should be better.

Bouissou [12] highlighted that the possible flaw with all of the heuristics suggested is due to their lack of theoretical foundation. The paper highlights a new theoretical relationship between BDD size and modules of a fault tree. One possible constraint for any ordering heuristic using this method is that it should group the variables of a module. Using this modular property, the problem of determining a global optimal ordering can be split into smaller problems: determining optimal orderings for each module.

Freidman and Supowit [13] looked at the size of algorithms, in terms of computation time and number of operations required, currently used to produce a compact representation of the BDD. They present an ordering leading to the most compact representation. They present an algorithm with time complexity $O(n^23^n)$ an improvement over the previous best, which required $O(n!2^n)$, which is basically an algorithm applying brute force. Although the function $n^23^n$ grows very quickly with $n$ it does so dramatically more slowly than $n!2^n$. It has advantages over trees with relatively small numbers of basic events. However, in reality, even an algorithm of this size is not practical for industrial applications, where the number of variables may be hundreds rather than tens.

In summary, despite there being a considerable set of possible heuristics, it is not possible to determine how well they will perform for a given fault tree structure. Due
to the radically different nature between some fault trees it seems unlikely that a single simple heuristic will be enough to cope with producing the best ordering for all possible trees. A new approach needs to be investigated, either an approach which uses a combination of these heuristics determined by characteristics of the fault tree as to which one is best, or a completely new approach.

RULE BASED APPROACHES

A recent new approach to tackling the variable ordering dilemma is to use a rule based pattern recognition approach. There are several different types of pattern recognition approach, for example, classifier systems, neural networks, Bayesian methods and Fuzzy Logic. In the literature [14], the classifier system has been used in conjunction with a genetic algorithm. The purpose is to try and generate, initially randomly and then refine through training of the classifier system, a set of rules. Thus, when the classifier system is given a specific fault tree structure it can select, depending on the rules, the best of a set of alternative ordering schemes which will yield the most minimal BDD structure.

Classifier systems are a kind of rule-based machine learning system, with general mechanisms for processing rules in parallel, for testing the effectiveness of existing rules and for adaptive generation of new rules [17]. A classifier system, depicted schematically in figure 5, consists of three main components:

1) Rule and message system.
2) Apportionment of credit system.
3) Rule/message generation system.

Basically, the classifier system is trained on a data set of fault tree structures, with the characteristics of the fault tree structure coded as the input messages of the classifier system, and the known best ordering scheme is coded as the outputs of the system. Therefore, through a matching process a set of rules (classifiers) are generated depending on the relationship of these training inputs and outputs. New rules (classifiers) are injected into the system by using a genetic algorithm.
The research involved using fault tree structures from industry and also by random production using a computer program. Each tree structure was expressed in terms of the chosen characteristics and analysed prior to training to establish the best ordering scheme to produce the most efficient BDD representation. To determine the validity of the approach a set of six scheme alternatives were used [7]. The best scheme was identified by the minimum number of nodes in the BDD structure before minimization (removal of redundant nodes). This represented the efficiency of the BDD as the number of nodes is a good indication of the number of calculations required to quantify the diagram.

To evaluate the performance of the learning classifier system a test set of data was produced with different tree structures and known best ordering schemes. The performance is evaluated by comparing the number of correct scheme outputs predicted.

Results for the number of correct scheme predictions for the test data were encouraging. For the smaller fault tree structures the results have been more accurate. It was thought that the significant characteristics chosen to represent the fault tree structure may for the larger fault trees require more research. More tests need to be undertaken to provide further evaluations. With slight alterations to the set of factors chosen to adequately represent the complexity of the fault tree structure, and also by increasing the size of the training data set, it is anticipated that the results for larger trees will be as convincing as that currently shown for the smaller sized fault trees.
Using this classifier method the six ordering permutations possible each follow distinct heuristics. These heuristics are inflexible in that the tree must be traversed level by level or across from left to right (or permutations of both) in an ordered pattern. This may be a disadvantage as it precludes jumping around from one branch to another branch, which may offer a better ordering.

**ALTERNATIVE NEW IDEAS**

The authors have found a different method, which when tested on a large data set of example fault tree structures, outperformed the best of a selection of six alternative ordering schemes in producing a minimal (or near minimal) BDD structure. The method involved using the structural importance measure of each variable in the fault tree structure.

For each component it’s importance measure signifies the role that it plays in either causing or contributing to the occurrence of the top event. A numerical value is assigned to each basic event which allows it to be ranked according to the extent of its contribution to the occurrence of the top event.

Importance measures can be categorised in two ways: (1) deterministic; and (2) probabilistic. In this research a deterministic measure of importance was used since it is only the contribution of the component within the structure which can influence its ordering not its likelihood of occurrence. Deterministic measures assess the importance of a component to the system operation without considering the component’s probability of failure. One such measure is the structural importance measure, which is defined for a component \( j \) as

\[
IMP_j = \frac{\text{number of critical system states for component } j}{\text{total number of states for the } (n-1) \text{ remaining components}}
\]

A critical state for component \( j \) is a state for the remaining \( n-1 \) components such that a failure of component \( j \) causes the system to go from a working state to a failed state [18].

To illustrate this measure, consider the fault tree drawn in figure 6. The logic expression for the top event is:

\[
TOP = A + B.C
\]
The structural importance measure for component A ($IMP_A$) is calculated using table 1:

**TABLE 1**  
Possible Critical States For Component A

<table>
<thead>
<tr>
<th>All Possible States for other components</th>
<th>Critical State for A</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B, C$</td>
<td>Yes</td>
</tr>
<tr>
<td>$\overline{B}, C$</td>
<td>Yes</td>
</tr>
<tr>
<td>$B, \overline{C}$</td>
<td>Yes</td>
</tr>
<tr>
<td>$\overline{B}, \overline{C}$</td>
<td>No</td>
</tr>
</tbody>
</table>

(NB. the $\overline{B}$ means component B failure, etc.)

Hence $IMP_A = \frac{3}{4}$.

The structural importance measure of component B ($IMP_B$) is calculated using table 2. Therefore, $IMP_B = \frac{1}{4}$. Similarly, by structural symmetry the importance measure of component C, $IMP_C = \frac{1}{4}$.
In establishing the effectiveness of ordering components according to their structural importance in a fault tree to yield a minimalistic BDD structure the tabular approach presented above was not a practical proposition. Research by Lambert [19] found that using Birnbaum’s probabilistic measure of importance, with stated probabilistic values of failure for each component, the structural importance measure was produced.

Birnbaum’s measure of importance or criticality \( G_i(q) \) is defined as:

\[
G_i(q) = Q(1_i, q) - Q(0_i, q)
\]

where \( Q(q) \) is the probability that the system fails, and

\[
Q(1_i, q) = (q_1, q_2, \ldots, q_{i-1}, 1, q_{i+1}, \ldots, q_n)
\]

and

\[
Q(0_i, q) = (q_1, q_2, \ldots, q_{i-1}, 1, q_{i+1}, \ldots, q_n).
\]

From Lamberts paper it states that if we let \( q_j(t) \) (the probability of failure of component \( j \)) equal to \( \frac{1}{2} \) for all \( j \neq i \), then the fraction of possible states in which component \( i \) is critical, denoted by \( B_i \), is:

\[
B_i = \{ Q(1_i, \frac{1}{2}) - Q(0_i, \frac{1}{2}) \}
\]

Birnbaum calls \( B_i \) the structural importance of component \( i \).

Implementing this (numerically) with using the fault tree shown in figure 5, we see that

\[
Q(q) = q_A + q_B q_C - q_A q_B q_C
\]

Calculating the structural importance measure for \( A \):

\[
Q(1_A, q) = 1
\]

\[
Q(0_A, q) = q_B q_C
\]

Therefore,

\[
G_A(q) = 1 - q_B q_C,
\]

and

\[
G_A(1/2) = 1 - \frac{1}{4} = \frac{3}{4}.
\]

### TABLE 2
Possible Critical States For Component B

<table>
<thead>
<tr>
<th>States for other components</th>
<th>Critical State for B</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A, \overline{C} )</td>
<td>No</td>
</tr>
<tr>
<td>( \overline{A}, \overline{C} )</td>
<td>No</td>
</tr>
<tr>
<td>( A, \overline{C} )</td>
<td>Yes</td>
</tr>
<tr>
<td>( \overline{A}, \overline{C} )</td>
<td>No</td>
</tr>
</tbody>
</table>
The same principle is then applied to components B and C. Placing these events in the order reflecting their importance contribution gives $A < B < C$ (where B and C are equal). Using this ordering produces a minimal BDD.

To compare this new ordering permutation with the six previously identified schemes (ref. [14], Andrews and Bartlett, 1998) each ordering permutation was generated and then the number of nodes of the BDD were calculated. It is the number of nodes (before minimisation of the BDD structure) that is used in the comparison process. The results are as follows:

After 225 trees tested (shown in table 3):

<table>
<thead>
<tr>
<th>Nodes in comparison to previous best</th>
<th>No. of trees</th>
<th>% of trees</th>
<th>Total $&lt;=$ to previous best</th>
</tr>
</thead>
<tbody>
<tr>
<td>= to previous best</td>
<td>76</td>
<td>33.8</td>
<td></td>
</tr>
<tr>
<td>$&lt;$ previous best</td>
<td>97</td>
<td>43.1</td>
<td>76.9%</td>
</tr>
<tr>
<td>$&gt;$ previous best</td>
<td>52</td>
<td>23.1</td>
<td></td>
</tr>
</tbody>
</table>

From this, it is concluded that in approximately 77% of all the trees tested, the structural importance ordering yields a BDD of equal or smaller dimension than the previous best scheme ordering.

Hence, it is concluded that the structural importance measure of each variable, used to produce an ordering of all the variables in the fault tree yields a better result overall than any one of the six different ordering methods used in testing. Also the structural importance technique produces as good or better results, in 3 out of 4 instances, than the known best (which may not always be able to be located without using such techniques as the classifier approach already mentioned).

The research has shown the value of an ordering approach based on the component’s structural importance. The difficulty remains in finding an efficient algorithm to find the structural importance measures from the fault tree structure directly. Further research into this ordering possibility is in using approximation methods to establish the
structural importance. Additional improvements in the ordering may possibly be found by considering different techniques for ordering matched variables.

**CONCLUSIONS**

The research has highlighted the need for an efficient variable ordering for practical implementation of the BDD methodology. Of the methodologies discussed in this paper the following conclusions can be drawn:

1. There are a number of heuristic approaches, with some approaches being more beneficial in establishing a minimal BDD structure for some fault trees than for others. A minimalistic BDD structure may possibly be found for a specific fault tree using a heuristic approach, but unfortunately there is no way of determining which heuristic would be best suited to the fault tree under evaluation.

2. As no one best ordering heuristic seems to exist for all fault tree structures, using the classifier approach to select the best option from a set of possible ordering alternatives has proven to be a good step forward. This method has enabled a heuristic to be selected based on the characteristics of the fault tree, hence possibly producing better results by being able to vary the ordering heuristic used.

3. Obviously, the best solution to the variable ordering dilemma would be to find one approach that would benefit all fault tree structures. Using the structural importance measure of the components in the fault tree has proven to produce smaller or equal BDD structures for 77% of cases, than choosing from a set of alternatives. Initial tests show that this method could be the best to date if an efficient algorithm could be found to generate the importance measures.

**REFERENCES**