Comparison of Two New Approaches to Variable Ordering 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, FTA, is one of the most commonly used techniques for safety system analysis. There can be problems with the efficiency and accuracy of the approach when dealing with large tree structures. 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 BDD 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. There are a number of variable ordering heuristics in the literature, however the performance of each depends on the tree structure being analysed. These heuristic approaches do not yield a minimal BDD structure for all trees, some approaches generate orderings that are better for some trees but worse for others.

Within this paper two approaches to the variable ordering problem have been discussed. The first is the pattern recognition approach of neural networks, which is used to select the best ordering heuristic for a given fault tree from a set of alternatives. The second examines a completely new heuristic approach of using the structural importance of a component to produce a ranked ordering. The merits of each are discussed and the results compared.

1. Introduction

Over the past five years an alternative technique, to Kinetic Tree Theory (Vesely[1]), known as the Binary Decision Diagram (BDD) method[2-6] has been developed to analyse the fault tree. 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. 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. Lack of this efficient ordering for any tree structure is probably the reason that only one commercially available code[8] has been produced which is based on this method. The latest research looks at rule based approaches[9] to identify an ordering scheme that yields an efficient ordering of the fault tree variables. This paper compares two new approaches to the variable ordering problem. The first uses the rule based approach of neural networks to select the best ordering heuristic for a given fault tree from a set of alternatives. The second approach uses the structural importance of each of the basic events of the tree to produce a ranked ordering.

2. 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.

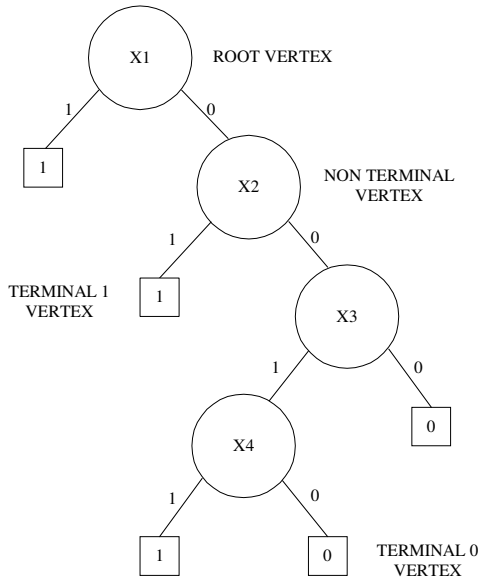

Figure 1: A Binary Decision Diagram.

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) X1 2) X2 3) X3X4.

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. [3,7]).

3. 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 closer it becomes to the optimal diagram. To illustrate this fact, consider the simple fault tree shown in figure 2. The tree has four basic events, where X1 is repeated.

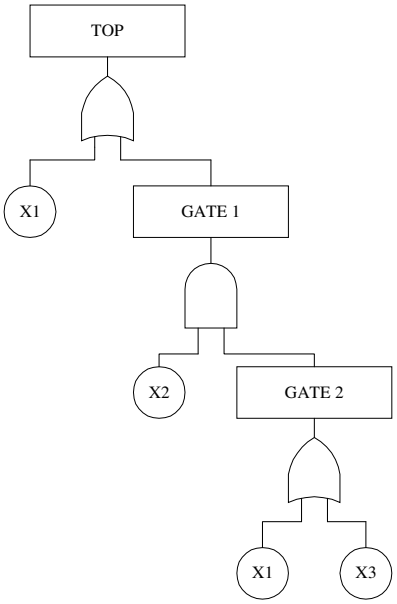

Figure 2: A simple fault tree.

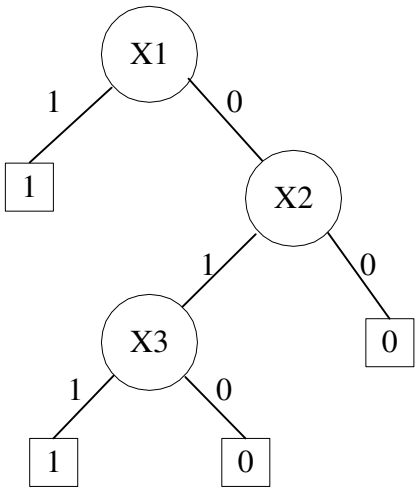

Figure 3: Ordering X1<X2<X3

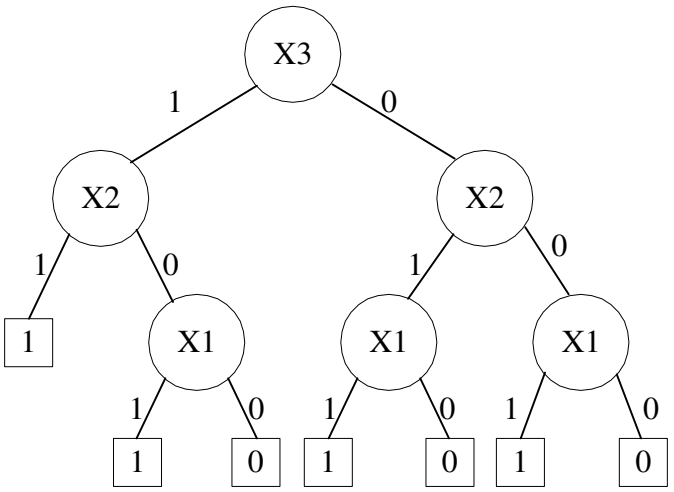

Figure 4: Ordering X3<X2<X1

If the basic events are taken in the order X1 then X2 and then X3, written in BDD notation as the ordering permutation X1<X2<X3, the resulting BDD is shown in figure 3. This structure consists of only three nodes, it is a minimal structure and hence produces only minimal cut sets.

However, if the alternative ordering permutation of X3<X2<X1 is taken the resulting BDD (shown in figure 4) consists of six 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 not be able to be produced within the computers memory capacity.

The objective would be to produce an ordering scheme that achieves the 'best' BDD for any given fault tree structure as currently the resulting BDD in the conversion process varies considerably depending on which ordering heuristic is chosen. The remaining sections of this paper will discuss two new approaches to tackling the ordering problem in terms of trying to find a method that will yield a minimal BDD for all fault trees. The first looks at finding the best solution from a set of alternatives. Similar work has been carried out using genetic algorithms[11] and the new approach highlighted in section 4 extends this work by considering a different pattern recognition methodology of neural networks. The second approach investigated by the authors looks at a completely new heuristic using structural importance measures (section 5).

4. Using Neural Networks to Select an Ordering Scheme from a Set of Alternatives

4.1 The Neural Network Approach

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[9], the classifier system has been used in conjunction with a genetic algorithm. Results for the number of correct scheme predictions for the test data were encouraging and predictions were better for smaller trees. Additional work was required to gain the same accuracy for larger tree structures. Utilising the evident potential of the pattern recognition approach, the research is extended by trying a second method of neural networks, which may be more suited to this problem.

The neural network is another method of identifying patterns. It can be regarded as a particular choice for a set of functions that map a set of input variables to a set of output variables. There are a number of different neural networks and the multi-layer feedforward network or multi-layer perceptron has been applied. The network is made up of a series of layers with connections running from every unit in one layer to every unit in the next layer. These connections are known as the *weights* and they control the influence each node has on propagating the intermediate outcome to the output nodes. Typically the network consists of a set of input nodes that constitute the input layer, one or more hidden layers of nodes, and an output layer of nodes (as shown in figure 5).

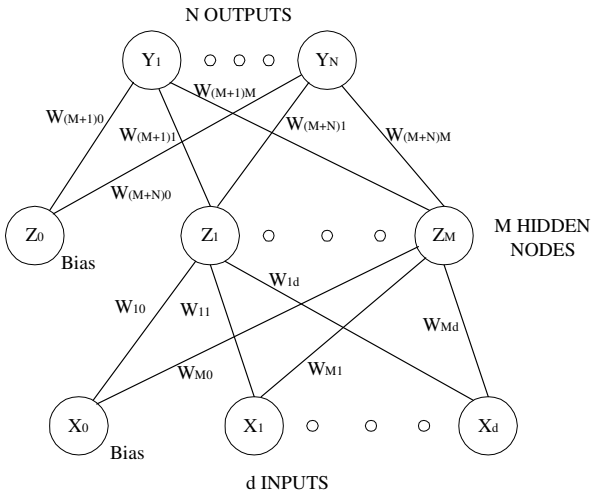

Figure 5: Diagram of a multi-layer perceptron.

There are two modes of operation: a training phase to determine optimum weights of network and a predictive phase to generate the desired outputs for a previously unseen input. During the training phase multi-layer perceptron commonly uses an algorithm known as the error back-propagation algorithm. The algorithmic process consists of two possible passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, an input vector is applied to the input nodes of the network, and subsequent outcomes are evaluated layer by layer. Hidden layer node values, $v_j(n)$ are calculated by taking the linear combination between the hidden node j and weight connections w_{ji} to each input $x_i(n)$. A non-linear activation function, $g(\ )$, is applied to the linear combination that is calculated.

$$v_j(n) = g\left(\sum_{i=0}^{d} w_{ji}.x_i(n)\right) \tag{1}$$

The values for the output layer nodes, $y_k(n)$ are established by evaluating the linear combination calculated between output node k and weight connection w_{kj} to each hidden node $v_j(n)$. An activation function g*( ) is then applied to the result, this may be the same function as used in the hidden layer or different.

$$y_k(n) = g^*\left(\sum_{j=0}^{m} w_{kj}.v_j(n)\right) \tag{2}$$

The activation function commonly used is the sigmoidal function, given by equation 3, where a is the linear combination output.

$$g(\mathrm{a}) = 1 / (1 + exp\ (-a)\ ) \tag{3}$$

During the backward pass, the weights are all adjusted in accordance with an error-correction rule. This rule is applied at the output nodes for each training pattern n, and takes the target response of each node t_j, and subtracts from it the response generated for that node by the network y_j, to produce an error e_j.

$$e_j(n) = t_j(n) - y_j(n)$$

This error signal is then propagated backward through the network, against the direction of weight connections, hence the name "error back-propagation". The weights are adjusted so as to make the actual response of the network move closer to the desired response.

The training phase involves a number of cycles whereby on each cycle the search for better weights is directed to a new area as defined by a specified search parameter. When the error has been reduced sufficiently it is these weights that are used as fixed values in the predictive phase. How well the network has been trained and models the problem will be reflected in the prediction of new input data. If the network has been trained well it will generalise well to new data and a correct response should be predicted.

To commence the training phase of the network, the number of inputs and outputs of the problem need to be determined and the training data set constructed. Sections 4.2-3 review the inputs and the outputs for this variable ordering problem, and section 4.4 the data sets for training, and test set to assess the predictive capability of the trained network.

4.2 Preferences for the Variable Ordering Schemes

The objective of the initial neural network modelling will be to establish the capability of the neural network to select the best ordering heuristic from a restricted group of alternatives for a given fault tree. In this study 6 different potential structured ordering schemes are used and referred to as:

- *Top-down, left-right approach:*
  - is produced by listing the variables in a top-down, left-right manner from the original fault tree structure.
- *Modified top-down, left-right approach:*
  - as the top–down, left-right approach but repeated events along each level are considered first.
- *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.
- *Modified depth-first approach:*
  - As unmodified version but with repeated events considered first.
- *Priority depth-first approach:*

- takes the depth-first approach one step further and considers subtrees with only basic event inputs first.

- *Modified priority depth- first approach:*
  - consider repeated events first in the ordering.

More details of these ordering schemes can be found in reference [8].

4.3 Fault Tree Characteristics

A difficulty in the neural network approach is in characterising the problem at hand. Some fault tree attributes have been selected to characterise each tree. The input layer of the neural network are variables which represent the eleven characteristics which were selected to represent the fault tree structure, and the output layer of nodes within the network are used to model the six scheme preferences.

To summarise, the characteristics that have been chosen to represent the fault tree structure are:

- *Percentage of AND gates* – examining the distribution of AND gates compared with OR gates;
- *Percentage of different events repeated* – calculated by examining the number of different events within the tree and finding the percentage of which were repeated;
- *Percentage of total events repeated* – calculating the percentage of repeated events in the set of all events in the tree (including repetitions);
- *Top gate type* – distinguishing between an OR or AND gate as the top gate type;
- *Number of outputs from top gate* – this is the number of branches from the top gate;
- *Number of levels of tree* – defines the depth of the tree;
- *Number of basic events*;
- *Maximum number of gates in any level* – examining each level and finding the level which has the highest number of gates;
- *Number of gates with gate outputs only* – counting the number of gates which only have branches going to gates, not events as well;
- *Number of gates with event outputs only* - counting the number of gates which only have branches going to events, not gates as well;
- *Highest multiple of a repeated event* – finding the number of occurrences of the most repeated event.

4.4 Training and Test Sets

To generate an appropriate neural network model and to test its performance a training and test set of fault trees for the problem were required. Fault tree structures used were taken from industry and randomly generated using a computer. The total in the training set was 205 and 20 test trees in the test set. Each fault tree was examined for the appropriate characteristics, and the BDD for each ordering permutation was constructed to evaluate which ordering permutation produced the smallest BDD. This

information, both inputs and outputs, could then be used to train the neural network to recognise the pattern or relationship between the fault tree and the best ordering heuristic to give a minimal BDD.

To evaluate the performance of the neural network a test set of data was produced with different tree structures and known best ordering schemes. The number of correct scheme preferences the network predicted evaluated the performance of the network.

4.5 Results of Most Appropriate Network Constructed

A number of models have been trained and tested with varying results. However, the best networks that have been constructed using the multi-layer perceptron approach have been very promising. The best network architecture is given in table 1. A single hidden layer was used which comprised of five nodes. The enhanced gradient descent algorithm was used to optimise the weights.

The weights generated in training and used in the prediction phase for previously unseen data are given in table 2 and 3. Table 2 shows the weights which connect the input nodes to each of the 5 hidden nodes, and table 3 indicates the weights from the hidden layer to the output nodes.

Structure Considerations	
Number of Input Nodes	11
Number of Output Nodes	6
Number of Hidden Layers	1
Number of Nodes in Hidden layer(s)	5

Table 1: Best Network Architecture

	Hidden 1	Hidden 2	Hidden 3	Hidden 4	Hidden 5
Bias node	0.147330	0.243064	0.539561	0.673099	-0.331514
Input 1	0.641682	0.183194	-0.924516	-0.665176	0.708712
Input 2	0.796946	0.886754	0.186239	-0.306955	0.086139
Input 3	0.842000	0.883227	0.623115	-1.054132	-0.037604
Input 4	0.382882	0.365247	-0.433369	-0.995933	0.156702
Input 5	0.506398	0.347366	-0.710282	-0.981350	-0.983196
Input 6	0.642397	0.202210	1.478797	1.438517	0.256880
Input 7	-0.382380	0.972270	0.371361	-0.064075	-0.772999
Input 8	-0.963998	0.648210	0.894712	0.339276	-0.428676
Input 9	0.329570	0.264644	0.067410	-0.007906	0.446248
Input 10	-1.004186	0.236299	0.296611	0.159278	0.035732
Input 11	0.603500	0.120445	-0.361202	-0.926677	0.304544

Table 2: First Layer Weights of Best Network

	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6
Bias node	0.435191	0.034015	0.902266	0.184042	0.291198	-0.378771
Hidden 1	1.562687	0.156753	0.383328	0.803264	-0.271374	0.430375
Hidden 2	0.732578	0.507769	-0.219266	0.543601	0.895298	0.526418
Hidden 3	1.316444	-0.003383	-0.184228	-0.025922	0.851694	0.116150
Hidden 4	-0.566968	0.253515	0.102770	-1.121804	-1.121293	0.104950
Hidden 5	0.199515	0.194763	-0.108666	-0.156623	-0.405690	-0.617658

Table 3: Second Layer Weights of Best Network

Using this architecture, the network predicted 14 out of the 20 test trees with correct scheme preferences. The error at the end of training was 0.341187. Thus, when trying to establish the best scheme choice which will lead to the smallest BDD for a previously unseen fault tree the network will predict successfully the correct option on 70 % of occasions.

For the remaining 30 percent of predictions, the ordering heuristic produced would yield a BDD of non-minimal dimension. The range of deviation from the minimal varied depending on the fault tree. Some predictions selected heuristics which resulted in a BDD with only a few nodes greater than smallest, whereas others where considerably larger.

5. Rankings Based on Structural Importance

5.1.Problem Areas With Current Heuristics

On reviewing the heuristics currently in the literature certain problem areas were evident. One was that many of the heuristics were affected by how the fault tree was drawn, therefore for the same logic expression a number of different BDDs could result depending on how the fault tree was represented. Many heuristics do not allow for components to be selected from different branches of the tree and lie next to each other in the ordering list. Another problem is how to deal with matched components in methods that assign weights to each basic event.

From this the properties required in a good ordering heuristic seem to be:

- The contribution of an event to the system failure mode must be reflected in the ordering produced.
- The ordering must be robust i.e. the ordering must be dependent upon the logic function represented by the fault tree and not influenced by the way the fault tree has been drawn.
- To uniquely map the fault tree onto a single event ordering.

Considering these points the structural importance measure was investigated. This heuristic satisfies two out of the three points above. It does represent the contribution each component makes to the occurrence of the top event, and it is also unaffected by the way the tree is written or drawn. However, the ordering produced is not unique

because ties may result with some of the component measures and the means of breaking these ties will affect the ordering.

5.2 Structural Importance Explained

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.

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

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

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[10].

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

$$TOP = A + B \cdot C$$

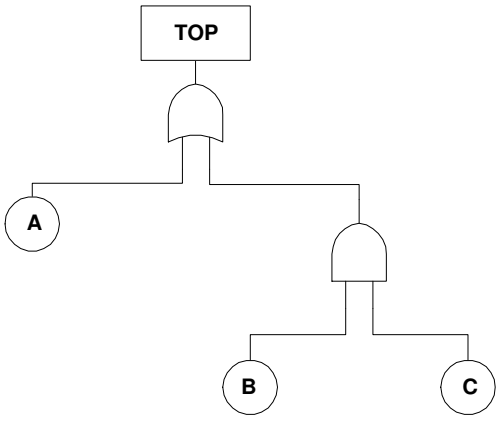

Figure 6: Simple fault tree structure

The structural importance measure for each component is given in table 2. The structural importance value for component A, IMP_A = ¾. The structural importance measure of component B (IMP_B) is equal to ¼. Similarly, by structural symmetry the importance measure of component C, IMP_C = ¼.

All Possible States for components B and C	Critical State for A	All Possible States for components A and C	Critical State for B
B, C	No	A, C	No
$\overline{B}, C$	Yes	$\overline{A}, C$	Yes
$B, \overline{C}$	Yes	$A, \overline{C}$	No
$\overline{B}, \overline{C}$	Yes	$\overline{A}, \overline{C}$	No

(NB. the $\overline{B}$ means component B working)

Table 2: Critical States for Component A and B

In establishing the effectiveness of ordering components according to their structural importance to yield a minimalistic BDD structure the tabular approach presented above was not a practical proposition. Lambert[10] 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 ($G_i(\boldsymbol{q})$) is defined as:

$$G_i(\boldsymbol{q}) = Q(1_i, \boldsymbol{q}) - Q(0_i, \boldsymbol{q})$$

where $Q(1_i, \boldsymbol{q}) = (q_1, q_2, \ldots, q_{i-1}, 1, q_{i+1}, \ldots q_n)$ and is the probability that component i fails and the system fails, and $Q(0_i, \boldsymbol{q}) = (q_1, q_2, \ldots, q_{i-1}, 0, q_{i+1}, \ldots q_n)$, which is the probability that component i is functioning and the system fails.

From Lamberts paper it states that if we let $q_j(t)$ (the probability of failure of component j) equal to ½ for all $j \neq i$, then the fraction of possible states in which component i is critical, denoted by B_i, is:

$$B_i = \{ Q(1_i, ½) - Q(0_i, ½) \}$$

Birnbaum calls B_i the structural importance of component i.

Implementing this (numerically) using the fault tree shown in figure 6, we see that the probability of the top event occurrence, $Q(\boldsymbol{q})$ is:

$$Q(\boldsymbol{q}) = q_A + q_B q_C - q_A q_B q_C$$

Calculating the structural importance measure for A;

$$Q(1_A, \boldsymbol{q}) = 1$$
$$Q(0_A, \boldsymbol{q}) = q_B q_C$$

Therefore, $G_A(\boldsymbol{q}) = 1 - q_B q_C$,
and $G_A(\boldsymbol{1/2}) = 1 - ¼ = ¾$.

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.

5.3 Application Of Structural Importance Approach

To compare this new ordering permutation with the six previously identified schemes (ref. [8], 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 that is used in the comparison process. The results after 225 trees were tested are shown in table 3.

Nodes in comparison to previous best	No. of trees	% of trees	Total =/<
= to previous best	77	34.2	
< previous best	96	42.7	76.9 %
> previous best	52	23.1	

Table 3: Results Using Structural Importance Measure For Ordering

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.

When considering each of the six ordering heuristics individually, the repeated event version of the top-down, left-right approach, produced the largest number of 'the smallest' BDDs in comparison to the other heuristics. Using this heuristic the smallest BDD was produced on 29.8 % of occasions. Hence, it is concluded that using the structural importance measure of each variable to produce an ordering yields a better result overall than any one of the six different ordering methods used in testing.

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.

The best method currently seems to be to use the neural network approach to select the best ordering heuristic for a given fault tree, and use this to construct the BDD. From this BDD the structural importance values can be produced to generate the second BDD from which the analysis procedure is to be carried out.

6. Conclusions

Both methods are feasible in improving the BDD ordering problem, although both use indirect methods to find the solution, for example the neural network needs to be

trained on the problem before it can be used, and the structural importance measures require the BDD to be generated first. To improve the methods the performance of the neural network needs to exceed 14 and near 20 correct predictions for the test data set. The structural importance technique is required to be generated from the tree directly rather than using two BDDs. Extra work needs to be carried out in both areas to make the two approaches ideal for the ordering problem.

7. References

1. W.E.Vesley, "A Time Dependent Methodology for Fault Tree Evaluation". *Nuclear Eng. and des.*, 13, 1970, p337-360.
2. S.B.Akers, "Binary Decision Diagrams". *IEEE Transactions on Computers*, vol. C-27, 1978, p509-516.
3. A.Rauzy, "New Algorithms for Fault Tree Analysis". *Reliability Engineering and System Safety*, vol. 40, 1993, p203-211.
4. R.M.Sinnamon and J.D.Andrews, "Quantitative Fault Tree Analysis Using Binary Decision Diagrams". *European Journal of Automation*, vol. 30, No. 8, 1996.
5. A.Rauzy, "A Brief Introduction to Binary Decision Diagrams". *European Journal of Automation*, vol. 30, No. 8, 1996.
6. R.M.Sinnamon and J.D.Andrews, "Fault Tree Analysis and Binary Decision Diagrams". *Proceedings of the Reliability and Maintainability Symposium*, Las Vegas, January 1996.
7. R.M.Sinnamon and J.D.Andrews, "New approaches to Evaluating Fault Trees". *Proceedings of Esrel'95 Conference*, June, 1995, p241-254.
8. Group Aralia, "Computation of Prime Implicants of a Fault Tree Within Aralia". *Proceedings of the European Safety and Reliability Association Conference, ESREL'95*, 1995, p190-202.
9. J.D.Andrews and L.M.Bartlett, "Efficient Basic Event Orderings for Binary Decision Diagrams". *Proceedings of the Annual Reliability and Maintainability Symposium*, Anaheim, 1998, p61-68.
10. H.E.Lambert, "Measures of Importance of Events and Cuts Sets in Fault Trees", *Reliability and Fault Tree Analysis: Theoretical and Applied Aspects of System Reliability and Safety System Assessment*. SIAM, Philadelphia, 1975, p77-100.
11. L.M.Bartlett and J.D.Andrews. "Efficient Basic Event Ordering Schemes for Fault Tree Analysis". Quality and Reliability Engineering International, vol. 15, 1999, p95-101.