# Choosing a Heuristic for the “Fault Tree to Binary Decision Diagram” Conversion, Using Neural Networks

Lisa M. Bartlett and John D. Andrews

***Abstract*—Fault-tree analysis is commonly used for risk assessment of industrial systems. Several computer packages are available to carry out the analysis. Despite its common usage there are associated limitations of the technique in terms of accuracy and efficiency when dealing with large fault-tree structures. The most recent approach to aid the analysis of the fault-tree diagram is the BDD (binary decision diagram). To use the BDD, the fault-tree structure needs to be converted into the BDD format. Converting the fault tree is relatively straightforward but requires that the basic events of the tree be ordered. This ordering is critical to the resulting size of the BDD, and ultimately affects the qualitative and quantitative performance and benefits of this technique. Several heuristic approaches were 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. There is no single heuristic that guarantees a minimal BDD for any fault-tree structure. This paper looks at a selection approach using a neural network to choose the best heuristic from a set of alternatives that will yield the smallest BDD and promote an efficient analysis. The set of possible selection choices are 6 alternative heuristics, and the prediction capacity produced was a 70% chance of the neural network choosing the best ordering heuristic from the set of 6 for the test set of given fault trees.**

***Index Terms*—Binary decision diagrams, fault-tree analysis, neural networks, variable ordering heuristics.**

## ACRONYMS[1]

| | |
|---|---|
| BDD | binary decision diagram |
| FT | fault tree |
| MLP | multi-layer perceptron |
| NN | neural network |
| RBF | radial basis function. |

## I. INTRODUCTION

### *Definition*

PERCEPTRON: A single-layer neural network whose weights and biases can be raised to produce a correct target vector when presented with the corresponding input vector.

FT analysis provides a diagrammatic description of the causes of system failure in terms of component failures. It permits both a qualitative and quantitative assessment of the failure mode. Problems are encountered in terms of efficiency and accuracy of the analysis process when large FT structures are investigated. To overcome these limitations a new technique, BDD [1]–[5], has been developed. This approach transforms the FT into an alternative representation, from which the failure combinations responsible for system failure (minimal cut sets) can be obtained efficiently and the exact failure probability can be calculated directly from the new representation.

To use the BDD, the FT is transformed by considering each of the basic events of the tree in a specified order. This ordering is crucial for the resulting size of the BDD, and hence for its benefits. The smaller the BDD, the less minimization needs to occur to find the minimal cut sets of the tree; also the quantification process involves fewer stages.

Several research papers investigate various ordering strategies and heuristics [6]–[12]. From the research to date, several heuristics have been developed which are effective for specific FT structures; but a general heuristic that produces a minimal BDD for all FT is not available. Instead of developing a new heuristic, the advantages of the current heuristics have been used. The aim of this research was to find the best ordering heuristic within a group of possibilities for a specific tree, thus allowing variation for different trees. To find the most appropriate heuristic a NN approach to finding patterns has been investigated.

### *Perspective*

Examining the set of 6 ordering heuristics used in the study, found that the scheme that produced the smallest BDD on the largest number of occasions was the modified top–down, left–right approach. The predictive fraction of producing a minimal BDD was 29.8%. This research showed that using this new NN, which allows for selecting an ordering heuristic from a set, had a predictive capability of 70%. The remaining 30% of responses varied across the various NN models. For the MLP network, incorrect predictions on the test data-set ranged from being only a few nodes larger than the optimal BDD to being a maximum of 50 nodes larger. For the RBF network, in the majority of test trees, 5 out of the 6 incorrect predictions were the second best choice, thus the BDD was only slightly larger than the smallest, and hence would have made very little difference to the analysis. This research suggests that a suitable “mechanism” for finding the appropriate ordering heuristic for a given FT to promote an efficient conversion process to the BDD has been found. The NN is a novel technique to help

Manuscript received May 19, 2000; revised May 3, 2001. Responsible Editor: S. Rai.

The authors are with the Department of Mathematical Sciences, Loughborough University, Loughborough, LE11 3TU, UK (e-mail: L.M.Bartlett@lboro.ac.uk; J.D.Andrews@lboro.ac.uk).

Publisher Item Identifier 10.1109/TR.2002.802892.

[1]The singular and plural of an acronym are always spelled the same.

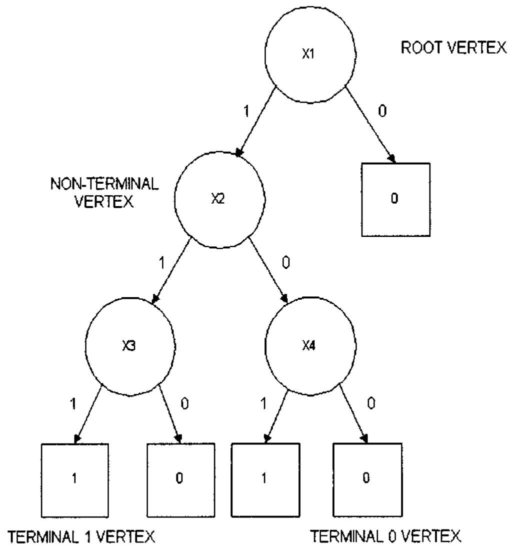

Fig. 1. A binary decision diagram.

in this ordering problem with important improvements (40%) in producing a minimal BDD in comparison to the previous best of the modified top–down, left–right approach. Now that NN is proven to be useful, it warrants further investigation to improve upon the predictive ability. The characteristics of the FT should be scrutinized for their relevance in determining the best ordering heuristic to use; and the way the tree is drawn needs to be addressed.

## II. Binary Decision Diagrams

A BDD is a directed acyclic graph, as shown in Fig. 1. All paths through the BDD begin at the root vertex and terminate in 1 of 2 states:

- a **1** state (system failure), or
- a **0** state (system success).

A BDD is composed of terminal and nonterminal vertices, which are connected by branches. Non-terminal vertices correspond to the basic events of the FT.

All the left branches leaving a vertex are the **1** branches (component failure occurs); all the right branches are the **0** branches (component functional). Every path begins 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 FT. For example, the cut-sets of Fig. 1 are:

1) X1 X2 X3    2) X1 X4.

The method to convert a FT to its equivalent BDD is described in many publications; refer to them for details [3].

## III. Variable Ordering

### A. The Problem

In constructing the BDD, the ordering of the basic events is crucial to the size of the resulting diagram. An inefficient ordering scheme produces a nonminimal BDD structure. Alternative ordering schemes produce BDD of various sizes; the smaller the BDD, the more optimal the diagram. To illustrate this, consider the simple FT in Fig. 2. The tree has 4 basic events, where X2 is repeated.

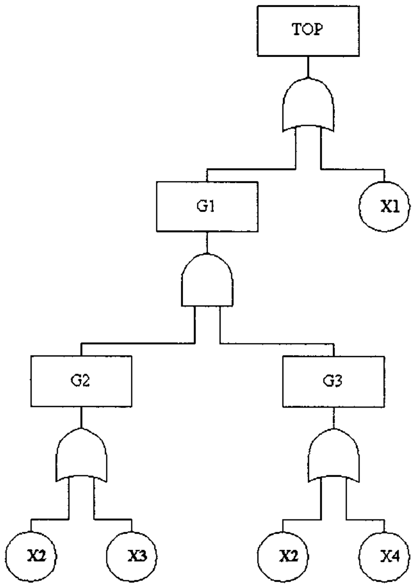

Fig. 2. A simple fault-tree.

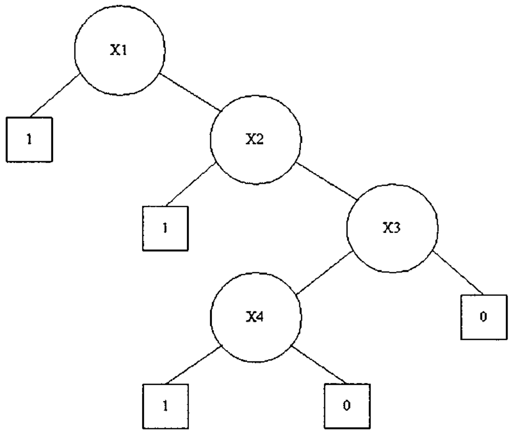

Fig. 3. Ordering X1 $<$ X2 $<$ X3 $<$ X4.

If the basic event ordering permutation of X1 $<$ X2 $<$ X3 $<$ X4 is taken, then Fig. 3 shows the resulting BDD. This structure consists of only 4 nodes; it is a minimal structure and hence produces only minimal cut sets.

However, if the alternative ordering permutation of X4 $<$ X3 $<$ X2 $<$ X1 is taken, then Fig. 4 shows the resulting BDD; it consists of 7 nodes, is nonminimal, and yields nonminimal cut sets. Analyzing the second BDD involves a minimization procedure to find the minimal cut sets; the quantification process would require additional steps compared to analyzing the BDD in Fig. 3. For larger FT structures, the efficiency of the resulting BDD is more critical, and in the worst case of using a poor ordering permutation, the diagram might be unsolvable.

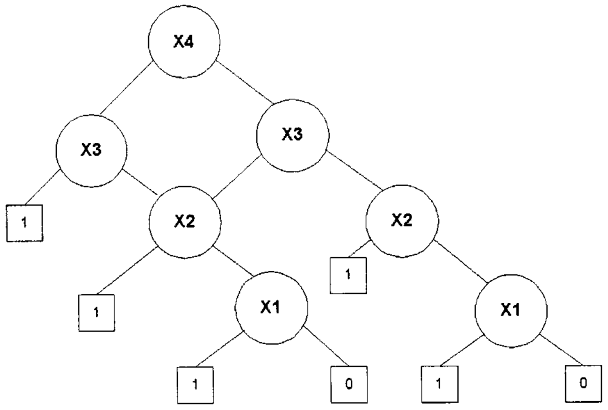

Fig. 4. X4 < X3 < X2 < X1.

The objective is to produce an ordering scheme that converts the FT to the smallest BDD possible, to take full advantage of the benefits of this approach. Instead of finding a completely new heuristic, the heuristics already in the literature have the potential to produce a minimal BDD if the appropriate heuristic can be found. Thus, a mechanism was needed to locate the appropriate heuristic for a specific tree. The remainder of this paper examines the pattern-recognition approach of NN which was used as the "mechanism."

## B. Heuristics Already Available

This ordering problem has been investigated by two research communities, electronic circuits and FT. The diverse nature of the logic functions used in these applications makes the validity of conclusions drawn from the circuits [8], [11], [12] questionable in terms of appropriateness for FT.

The research in [6] investigated the effects that various ordering schemes produce on the resulting size of the BDD for FT structures. Six heuristics were investigated; there were vast differences in the number of computations required to construct the BDD when each of the different orderings was used. Thus, these 6 heuristics are used to demonstrate the feasibility of the "mechanism" to select the best ordering-option. These are discussed in more detail in Section V.

## IV. Overview of Neural Networks

NN is a 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. Thus, NN could be applied to the variable-ordering problem, wherein the inputs are the FT and the outputs are the various ordering heuristics. There are different types of NN: single layer networks, multi-layer networks, and radial basis functions. The latter 2 networks are used in this research.

## A. Multi-Layer Perceptron

The multi-layer network (perceptron model) has several distinct layers. The network has an input layer, an output layer, and several hidden layers each with a specified number of nodes. Fig. 5 shows the general architecture of the network. Each of the layers is connected by weights, which determine the predictive ability of the network.

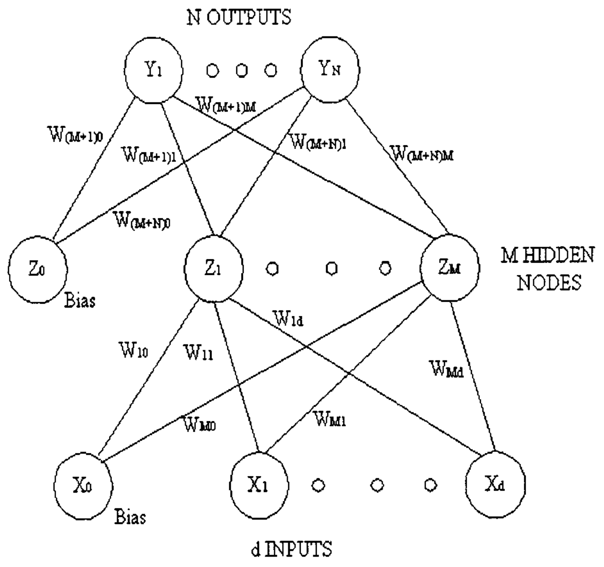

Fig. 5. Diagram representing a multi-layer perceptron.

There are two modes of operation:

- a training phase to determine the optimum weights of the network,
- a predictive phase to generate the desired outputs for a previously unseen input.

During the training phase, multi-layer perceptrons commonly use the error back-propagation algorithm. The algorithmic process consists of two possible passes through the various 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 are calculated by taking the linear combination between the specified hidden node and weight connections to each input. A nonlinear activation function is applied to the linear combination that is calculated.

The values for the output layer nodes are found by evaluating the linear combination calculated between the selected output node and weight connection to each hidden node. An activation function is then applied to the result, this can be the same function as used in the hidden layer, or can be different.

The forward calculation pass uses fixed weights. During the backward pass, on the other hand, the weights are adjusted in accordance with an error-correction rule or a delta rule. The error-correction rule, which is applied at the output nodes for each training pattern $n$, 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 $e_j$. 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 to make the actual response of the network move closer to the desired response. When the error has been sufficiently reduced, then 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" are reflected in the prediction of new input data. If the network has been trained well, then it will generalize well to new data, and a correct response should be predicted.

### *B. Radial Basis Functions*

The theoretical origins of radial basis functions are founded in the techniques for performing exact interpolation of a set of data points in a multi-dimensional space [13]. The approach introduces a set of basis functions which depend on the distance between the "set of input vectors" and a "prototype vector defined at the basis function center."

Like multi-layer perceptrons, the network has several input nodes, representing each feature component of the problem, and several output nodes, or targets. The radial basis function network typically has only 1 layer of hidden nodes, where each node has an activation function centered on a chosen radial basis function.

The connections between the "input layer" and the "hidden layer of radial basis functions" represent the vectors determining the centers of the radial basis functions. The first stage of training identifies these center parameters.

The connections between the basis functions and the output layer represent the weights of the network; these are determined during the training phase of the NN. The output representing the mapping of the radial basis function NN is then taken to be a linear combination of the basis functions and the weight vectors associated with all paths to the desired output node.

Training of the radial basis function NN can be considerably quicker than training a multi-layer perceptron, because only 1 cycle is performed. There are 2 distinct phases of training. In phase 1, the input data-set alone is used to determine the network parameters (the basis function centers and width or spread parameters). In phase 2, the network parameters remain fixed while the second layer weights are established.

## V. Modeling the Ordering Problem

The difficulty in the NN approach is in correctly modeling the problem. Some FT attributes have been intuitively selected to characterize the structure. The input layer of the NN represents the 11 characteristics which were selected to represent the FT structure; the output layer of nodes within the network are used to model the 6 ordering heuristic preferences.

The 11 characteristics chosen to represent the FT structure, and the reasoning behind their selection are summarized here:

1) Fraction of AND Gates: This indicates the balance of the tree in terms of AND/OR gate types.

2) Fraction of Different Events Repeated: This characteristic indicates the proportion of repeated events and nonrepeated events within the tree. The higher the proportion of repeated events the more chance of nonminimal cut sets.

3) Fraction of Total Events Repeated: Considering all the events in the tree (all repetitions included), this factor is selected on the same belief as in 2).

4) Top Gate Type: Two possible gates are considered in the research as the starting gate of the alternating sequence of gates within the tree structure: AND and OR. This start point can influence the scheme choice.

TABLE I
SUMMARY CHARACTERISTICS (SC) OF "TRAINING" AND "TEST" FT USING MIN/MAX VALUES

| SC | Min/Max Values | |
|---|---|---|
| | Training FT | Test FT |
| 1 | 20 / 70 | 31 / 70 % |
| 2 | 0 / 93.44 | 1 / 63 % |
| 3 | 0 / 57.69 | 3 / 32 % |
| 4 | 126 / 99 | 10 / 10 |
| 5 | 2 / 15 | 2 / 7 |
| 6 | 2 / 12 | 4 / 8 |
| 7 | 4 / 208 | 28 / 123 |
| 8 | 2 / 34 | 3 / 28 |
| 9 | 0 / 13 | 0 / 6 |
| 10 | 2 / 33 | 4 / 29 |
| 11 | 0 / 16 | 2 / 6 |

5) Number of Outputs from Top Gate: This characteristic indicates the breadth of the tree, and influences the number of subtrees.

6) Number of Levels in The Tree: This characteristic indicates the depth of the tree, illustrating the size of the subtrees. Larger subtrees lead to more variation in the ordering list.

7) Number of Basic Events: The number of basic events in the tree is believed to be one of the most variable characteristics and possibly one of the most influential in the ordering process.

8) Maximum Number of Gates in Any Level: The number of gates increases the number of levels within the tree, and hence the possible effect on the ordering, especially when subtrees were used.

9) and 10) Number of Gates with "Just Event" or "Gate Only Inputs": Both of these characteristics were included because 2 of the heuristics used these characteristics in their rules for ordering.

11) Highest Number of Repeated Events: This characteristic was selected because the higher the repetition of an event, then the more problems it could cause with the introduction of nonminimal cut sets.

The ordering heuristics chosen to be the alternatives used for selection are:

- Top–down, left–right approach;
- Depth-first approach;
- Priority depth-first approach;
- Repeated event versions of each of the above.

The details of each of these 6 ordering heuristics are in [14].

To train and test both the multi-layer perceptron and radial-basis function NN, a set of examples is required. FT structures were used from industry and randomly generated using a computer program. All trees were analyzed for the chosen 11 characteristics and for best ordering heuristic alternative (using the number of nodes in the diagram). Table I (column #2) shows the summary characteristics of the "Training" set of FT. The minimum and maximum values are given, apart from characteristic 4 where the total number of trees with AND/OR gates is given. The characteristic number corresponds to the preceding list of 11 characteristics.

To evaluate the performance of the NN, a test-set of data was produced with various tree structures and known-best ordering

TABLE II
PREDICTIONS MADE BY MULTI-LAYER PERCEPTRON NETWORK

A ≡ Correct Scheme Prediction, B ≡ Target Outputs

| | Scheme 1 | Scheme 2 | Scheme 3 | Scheme 4 | Scheme 5 | Scheme 6 |
|---|---|---|---|---|---|---|
| 1 | 0.670841$^{B}$ | 0.708799 | 0.624587 | 0.630142 | 0.039991 | 0.197493 |
| 2 | 0.656986 | 0.606928 | 0.869536$^{A,B}$ | 0.673463 | 0.028306 | 0.177636 |
| 3 | 0.645769 | 0.711674$^{A,B}$ | 0.583244 | 0.636326 | 0.040711 | 0.202445 |
| 4 | 0.659933$^{B}$ | 0.797716$^{A,B}$ | 0.693398 | 0.675154 | 0.025590 | 0.170580 |
| 5 | 0.573971 | 0.811258$^{A,B}$ | 0.685900 | 0.667436 | 0.026590 | 0.173944 |
| 6 | 0.621941 | 0.694048 | 0.701801 | 0.681526$^{B}$ | 0.019957 | 0.146142 |
| 7 | 0.694922 | 0.741734$^{A,B}$ | 0.530858 | 0.637807$^{B}$ | 0.037144 | 0.189081 |
| 8 | 0.597916 | 0.813902$^{A,B}$ | 0.630858 | 0.574592 | 0.039956 | 0.189021 |
| 9 | 0.660024 | 0.696672 | 0.695251$^{B}$ | 0.675251 | 0.021767 | 0.201825 |
| 10 | 0.562133$^{B}$ | 0.702872$^{B}$ | 0.772498$^{A,B}$ | 0.682485$^{B}$ | 0.021365 | 0.151141 |
| 11 | 0.664468 | 0.714235$^{A,B}$ | 0.590533 | 0.665586 | 0.026375 | 0.156339 |
| 12 | 0.467304 | 0.699394 | 0.736468$^{A,B}$ | 0.626732 | 0.021955 | 0.173894 |
| 13 | 0.670376 | 0.706792$^{B}$ | 0.836947 | 0.641116 | 0.037559 | 0.155332 |
| 14 | 0.668940 | 0.685262 | 0.731781$^{A,B}$ | 0.637571 | 0.036777 | 0.193716 |
| 15 | 0.629205 | 0.718603 | 0.839411$^{A,B}$ | 0.638912 | 0.037296 | 0.199559 |
| 16 | 0.672490 | 0.812086$^{A,B}$ | 0.698450 | 0.617895 | 0.25145 | 0.172281 |
| 17 | 0.569735 | 0.873788$^{A,B}$ | 0.631487 | 0.606637 | 0.035630 | 0.186709 |
| 18 | 0.611487 | 0.706060 | 0.632438 | 0.638318$^{B}$ | 0.037023 | 0.191469 |
| 19 | 0.663175 | 0.677231$^{B}$ | 0.718790 | 0.633537 | 0.036556 | 0.189347 |
| 20 | 0.597623 | 0.666039 | 0.640738 | 0.671700$^{A,B}$ | 0.024656 | 0.165997 |

heuristics. The number of correct scheme preferences predicted by the network quantified the performance.

## VI. PREDICTIVE RESULTS USING NEURAL NETWORKS

### *A. The Network Architectures*

Table II summarizes, for the multi-layer perceptron approach, the output values calculated by the network. The bold-values show the correct scheme predictions; the italic-values are the target outputs. Table II shows that 14 correct predictions were made. The incorrect predictions need to be examined to gain an insight into the full predictive capacity of the network. The results are reviewed in Section VI-C.

The test set of FT upon which both the networks were tested totaled 20. Table I (col #3) summarizes the input "Test" characteristics for all FT. For both networks, the number of input nodes totaled 11, and the number of output nodes totaled 6. The multi-layer perceptron performed best, with 1 hidden layer comprising 5 nodes; the radial basis function optimal structure was found using 4 centers. For both NN techniques, the number of correct predictions on the test set of FT was: 14 out of 20. This indicates that for this set of trees, each network had a 70% chance of selecting the ordering heuristic for a specific tree which would result in the smallest BDD.

### *B. Outputs Generated by Networks*

The same data set of FT was used for testing both network approaches. The predictive potential of each network was established, depending on the number of correct ordering heuristic choices that were made. The correct predictions required for the 20 test trees are:

Scheme: 1, 3, 2, 1 & 2, 2, 4, 2 &

4, 2, 3, 1–4, 2, 3, 2, 3, 3, 2, 2, 4, 2, 4

where test tree 1 has desired output using ordering method 1, test tree 2 has best result with ordering method 3, and so on.

TABLE III
FIRST-TO-THIRD BEST CHOICES OF SCHEME OPTION FOR TEST TREES

| Tree | Best | 2nd Best | 3rd Best |
|---|---|---|---|
| 1 | 1 | 2 | 4 |
| 2 | 3 | 2 | 1 |
| 3 | 2 | 4 | 1 |
| 4 | 1 / 2 | 3 / 4 | 6 |
| 5 | 2 | 1 | 3 / 4 |
| 6 | 4 | 3 | 2 |
| 7 | 2 / 4 | 1 / 3 | 6 |
| 8 | 2 | 1 | 4 |
| 9 | 3 | 1 / 2 / 4 | 5 / 6 |
| 10 | 1-4 | 5 / 6 | |
| 11 | 2 | 1 | 4 |
| 12 | 3 | 4 | 1 |
| 13 | 2 | 1 | 4 |
| 14 | 3 | 2 | 1 |
| 15 | 3 | 1 / 2 | 4 |
| 16 | 2 | 1 | 4 |
| 17 | 2 | 4 | 1 |
| 18 | 4 | 2 | 3 |
| 19 | 2 | 1 | 4 |
| 20 | 4 | 3 | 2 |

For the radial basis function NN, the test trees which were correctly assigned the ordering scheme which produced the minimal BDD were numbers 3–13, 16, 17, 19. The incorrect predictions are also reviewed in Section VI-C.

### *C. Review of Incorrect Responses*

To make further assertions concerning the predictive capacity of the studied NN architectures, it is necessary to look at the incorrect predictions. Table III gives the first, second, third best ordering heuristic choices for each of the test trees. Where 2 numbers occupy a cell, either of the schemes can be used.

Of the 6 incorrect predictions produced using the optimal network architecture for the multi-layer perceptron,

- 4 is the second best ordering heuristic,

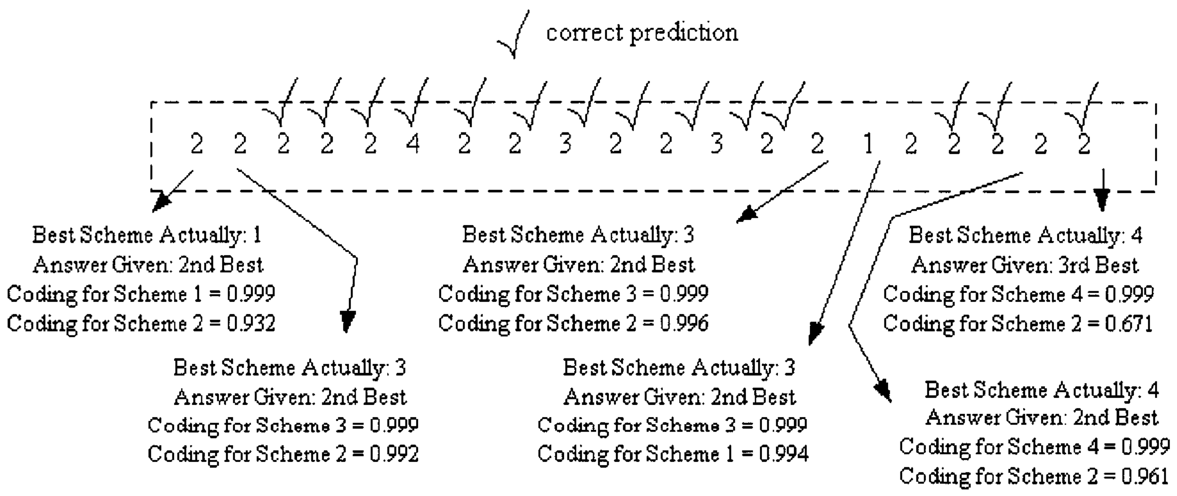

Fig. 6. Findings of the incorrect predictions.

- 1 is the third best ordering heuristic,
- 1 is the fourth best ordering heuristic.

On reviewing the size difference of the BDD produced using the second best ordering heuristic, the result is small, thus although not a "minimal BDD" a "near-minimal BDD" would have resulted using this ordering choice, which would have had little effect on the analysis procedure. The size difference for the 2 remaining incorrect schemes is slightly larger with between 10 and 50 nodes difference, but in both cases the analysis could still be carried out.

Examining the incorrect predictions made by the radial basis function NN, it became evident that a similar pattern emerged. Fig. 6 shows the predictions made with a summary given to the incorrect scheme choices; 5 out of the 6 incorrect predictions are near to best scheme coding. Looking at the values of the second-best option in comparison to the best, there is little difference in the coded values; thus the second-best option is not dramatically different, nor is the resulting BDD structure considerably larger than the optimal structure.

Hence, this NN technique can be used to solve the ordering problem, with optimal BDD being produced approximately 70% of the time for the given test set; for the remaining 30%, a near optimal BDD structure should be produced, hence facilitating an efficient analysis.

## References

[1] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Computers*, vol. C-27, pp. 509–516, 1978.
[2] A. Rauzy, "A brief introduction to binary decision diagrams," *European J. Automation*, vol. 30, no. 8, 1996.
[3] ——, "New algorithms for fault tree analysis," *Reliability Engineering and System Safety*, vol. 40, pp. 203–211, 1993.
[4] R. M. Sinnamon and J. D. Andrews, "Quantitative fault tree analysis using binary decision diagrams," *European J. Automation*, vol. 30, no. 8, 1996.
[5] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computers*, vol. C-35, no. 8, 1986.
[6] R. M. Sinnamon and J. D. Andrews, "Improved efficiency in qualitative fault tree analysis," in *Advances in Reliability Technology Symp.*, 1996.
[7] ——, "Fault tree analysis and binary decision diagrams," in *Proc. Reliability & Maintainability Symp.*, 1996.
[8] S. Minato, N. Ishiura, and S. Yajima, "On variable ordering of binary decision diagrams for the application of the multi-level logic synthesis," in *Proc. European Conf. Design Automation*, 1991, pp. 50–54.
[9] M. Bouissou, F. Bruyere, and A. Rauzy, "BDD based fault-tree processing: A comparison of variable ordering heuristics," in *Proc. ESREL 1997 Conf.*
[10] M. Bouissou, "An ordering heuristic for building binary decision diagrams from fault trees," in *Proc. Reliability & Maintainability Symp.*, 1996, pp. 208–214.
[11] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and improvements of Boolean comparison method based on binary decision diagrams," in *IEEE Trans. Computer Aided Design (Conference)*, 1988, pp. 2–5.
[12] M. Nikolskaia, "Binary decision diagrams and applications to reliability analysis," Doctoral dissertation, Univ. of Bordeaux, 1999.
[13] C. M. Bishop, *Neural Networks for Pattern Recognition*: Clarendon Press, 1995.
[14] J. D. Andrews and L. M. Bartlett, "Efficient basic event orderings for binary decision diagrams," in *Proc. Annual Reliability & Maintainability Symp.*, 1998, pp. 61–68.

**Lisa M. Bartlett** is a lecturer in the Department of Mathematical Sciences at Loughborough University. She gained a first class honors degree (1997) in mathematics and sports science, and Ph.D. (2000) in fault tree analysis methods from Loughborough University. Her Ph.D. research focused on the binary decision diagram approach, and her current research interests continue in this area.

**John D. Andrews** is a Professor in the Department of Mathematical Sciences at Loughborough University. He joined this department in 1989 having previously gained 9 years industrial research experience with British Gas and 2 years lecturing experience in the Mechanical Engineering Department at the University of Central England. His current research interests concern the assessment of the safety and risk of potentially hazardous industrial activities. This research has been heavily supported by industrial funding. Over recent years grants have been secured from the MOD, Rolls Royce Aero-Engines, Mobil North Sea, and Bechtel. Professor Andrews has numerous journal/conference publications along with a jointly authored book *Risk and Reliability Assessment* which is now in its second edition.