Journal of Applied Sciences 4 (2): 322-329, 2004
ISSN 1607-8926

Data Structures in Power System Reliability Estimation

Stella Morris and Morris A.G. Ezra
Faculty of Engineering, Multimedia University, Jalan Multimedia, 63100, Cyberjaya, Selangor, Malaysia

Abstract: In this paper a technique for the estimation of power system reliability using fault tree analysis (FTA) and the concept of data structures is presented. The proposed approach for representing the fault trees uses data structures concept and dynamic storage allocation which overcomes the dimensionality problem. This approach reduces the computational complexity and execution time. This method can efficiently handle systems having any number of cut-sets between the input-output nodes depending upon the available computer memory. Three sample power system networks are considered to test the efficiency of the algorithm developed and the computation times with different approaches are compared.

Key words: Power system reliability, loss of load probability (LOLP) index, loss of energy index, fault tree analysis (FTA), basic fault events, cut-sets, minimal cut-sets, data structures, dynamic storage allocation

INTRODUCTION

With the increasing complexity of power system configuration, the computation of reliability becomes quite tedious when one has to deal with a non series-parallel system. There has been a continuing and increasing interest in methods for power system reliability evaluation[1-3]. In the reliability analysis of complex systems, an usual approach is to describe the physical system by means of some logic diagram which expresses all functional relations between components or events considered to be elemental. Many methods are available for evaluating power system reliability[4-8]. The promising techniques which had already been derived include Heuristic methods, Simulation techniques, Minimal cut-set methods and Fault tree techniques.

A comparatively new and powerful approach to system reliability evaluation is Fault Tree Analysis. It is basically a systematic analysis of the system failure events and the subsystem and/or component failure events that can cause them. Fault trees represent system conditions symbolically and include all the basic fault events that may occur or expected to occur in the system. An appreciable amount of research has been devoted to find good methods for reliability evaluation in fault trees and networks[9-11].

This paper mainly emphasizes Fault Tree Analysis which has become a basic technique in the study of reliability for many types of systems. The way in which a fault tree is represented has a great influence in the amount of storage requirement and the computation time. However no method is known, which is more efficient and computes the probability of the top event in a desirable amount of time.

In conventional Fault Tree Analysis, the failure probabilities of components of a system are treated as exact values in estimating the failure probability of the top event. For many systems, it is often difficult to evaluate the failure probabilities of components from past occurrences because the environments of the system change. Furthermore it might be necessary to consider possible failure of components even if they have never failed before[12].

Even though many methods are available to represent the fault tree more efficiently in a computer that reduce the computing time for the operations that have to be performed with the fault tree, the concept of data structures[13] in representing the fault trees has reduced the computing time drastically. The main motivation of this algorithm is to obtain the minimal cut-sets as quickly as possible with the usage of the smallest amount of main core memory. Specifically, the main objectives of this paper work are detailed as follows:

i. Construction of fault tree by considering all system failure events.
ii. Representing the above constructed fault tree more effectively and efficiently (using data structures).
iii. Extraction of minimal cut-sets from the above represented fault tree.
iv. Calculation of overall system reliability by substituting the failure data on primary or basic events.

Corresponding Author: Stella Morris, Faculty of Engineering, Multimedia University, Jalan Multimedia, 63100, Cyberjaya, Selangor, Malaysia E-mail: stella.morris@mmu.edu.my

v. To implement parallel algorithm to reduce the computing time.

Power system reliability

Loss of load probability (lolp) index: LOLP method is employed in reliability studies of electric power systems to evaluate the probability that the system load might exceed the available generating capacity. LOLP can be defined as the probability of the system load exceeding the available generating capacity under the assumption that the peak load of each day lasts all day[14].

When the reliability of an electric power system is evaluated by the LOLP method, it is customary to take into account the uncertainty in the load forecasts by associating a peak-load distribution with the assumed daily peak-load duration curve.

Loss of energy index: Instead of LOLP index, another reliability measure (Loss of Energy Index) can be used for a generating system i.e., the ratio of the expected amount of energy not supplied during some long period of observation to the total energy required during the same period[14].

In some ways, the loss-of-energy index has more physical significance than the LOLP index. It is an 'in-depth' measure of reliability in that it will assume higher values for more serious events than for marginal failures even if their probabilities and frequencies are the same. On the other hand, the true loss of energy cannot be accurately computed on the basis of the cumulative curve of daily peaks. For this reason, the loss-of-energy index is seldom used. However, losses of energy are determined, using suitable load models, in several production cost evaluation programs.

Fault tree analysis: A comparatively new approach to system reliability evaluation is the fault-tree analysis(FTA). This technique has rapidly gained favors because of its versatility in degree of detail of complex systems[15-17]. Fault tree is one of the most commonly used representations of system logic.

The FTA starts by identifying an undesirable event, called top event associated with a system. In the case of system availability and reliability studies, this undesirable event is the non-operation of the system. The fault tree provides a diagrammatic representation of the event combinations resulting in the occurrence of this undesirable event. The basic steps in algebraic fault-tree analysis include alpha-numeric identification of gates and primary events, representation of the Boolean relationship (AND, OR) present at each gate, and the ordering of these appropriate to the subsequent analysis. It is made up of successive levels such that each event is generated out of lower-level events through various logic operators (AND and OR gates). This deductive process continues until one arrives at basic events which cannot be further decomposed i.e., mutually independent and probabilizable. These basic events can be failures, human errors, external conditions etc.

In general, fault trees are event oriented, whereas other methods are usually hardware oriented. The fault trees can be used to obtain minimal cut-set which define the modes of system failure and identify critical components. If the failure data on primary events is available, the reliability measures for the top event can be obtained.

The FTA has some distinct advantages. It provides a systematic procedure for documenting the cause-effect relationships between the failure as the various subsystem levels, and for identifying the most important failures and the weakest points in the system. Furthermore, they can pinpoint the system weakness in a visible form. This acts as a visual tool in communicating and supporting decisions based on the analysis and to determine the adequacy of the system design. To develop fault trees, the following basic steps are generally required:

i. Define the undesired event (top event) of the system under consideration.
ii. Thoroughly understand the system and its intended use.
iii. To obtain the predefined system fault condition causes, determine the higher order functional events. In addition, continue the fault event analysis to determine the logical interrelationship of lower level events that can cause them.
iv. After accomplishing steps, i to iv, construct a fault tree of logical relationships among input fault events. These are to be defined in terms of basic, identifiable, and independent faults.

Fault trees are particularly suitable aids in finding the cuts and minimal cuts of a system. Algorithms can be developed to accomplish the task. They are based on the fact that, as one proceeds downward from the top event, passing through an AND gate will invariably increase the size of a cut-set while passing through a OR gate will increase the number of these sets. A systematic search will produce a number of cuts. It is then important to identify all the minimal cuts. If the failure data on primary events is available, the reliability measures for the top event can be obtained.

Extraction of minimal cuts from fault trees: It is a tedious process since a computerized algorithm is required to obtain minimal cut-sets. Several algorithms have been proposed for obtaining the minimal cut-sets for the TOP event of a fault tree. The main goal of the proposed algorithm is to extract the minimal cuts with a minimum amount of computer memory requirement and with minimum computation time. The steps of the algorithm can be summarized as follows:

i. Resolution begins with the TOP event. If the TOP event is an AND gate, all inputs are listed as one set; if it is an OR gate, the inputs are listed as separate sets.
ii. Iterate until all OR gates with gate inputs and all AND gates are resolved. OR gates with only basic event inputs are not resolved at this time.
iii. Remove any super sets that still exist.
iv. Process any repeated basic events remaining in the unresolved OR gates. For each repeated event,
    a. The repeated event replaces all unresolved gates of which it is an input to form new sets.
    b. These new sets are added to the collection.
    c. This event is removed as input from the appropriate gates.
    d. Super sets are removed.
v. Resolve the remaining OR gates. All sets are now minimal cut-sets.

Once all the minimal cut-sets are identified, the system failure probability and in turn system reliability can be calculated from the knowledge of the failure data on primary events.

Fault trees using data structures: Due to complexity in the operation and maintenance of the power systems, even a sample power network of small size has large number of basic fault events. While representing a fault tree in the computer, dimensionality problem will arise. The size of the fault tree is different from system to system and also it varies for the same system at different operating conditions due to random occurrences of fault events in the useful life period. The available algorithms for evaluating fault tree logic use iterative search routines, Boolean reductions etc. Even when computer implemented, much manual effort is still required. It has been already proved that the application of data structures concept in many engineering problems has reduced the computing time drastically. The way a fault tree is represented impacts the amount of storage requirement. Using Data structures concept, the fault tree can be represented in computer memory as a tree structure in which the hierarchial relationship between the nodes is identified. This representation aids in efficient path tracking. Dynamic storage allocation is adopted for implementing the proposed algorithm.

Proposed algorithm

Tree structure representation: Structures are required to represent the hierarchial relationships between nodes. A tree structure can be efficiently used to cope with such a problem. Because tree structures are by definition complex, it is necessary that an efficient method of accessing the nodes of a tree has to be found.

Accessing the nodes of a tree

Pre-order traversal: This method has proved popular with the implementers of database system which use tree structures to represent the complex relationships. The tree represented in Fig. 1 is used as a test bed. The first node to be accessed is the root node A, followed by left most branch of the structure B which itself is a terminal node. With these two terminal nodes accounted for, the next branch node from the root accessed is C. The children of C are then accessed from the left to right, E and F. Finally, last branch node from the root is found, D, followed by its child, G, and its children from left to right, H and I which are the terminal nodes.

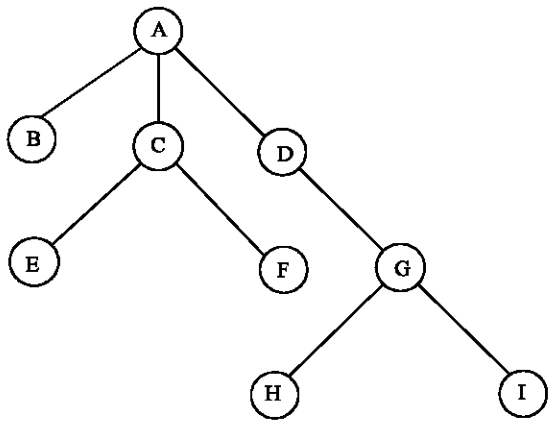

Fig. 1: Simple tree representation

Using this method, the nodes are accessed in the order:

A B C D E F G H I

The steps involved in the pre-order traversal are:

i. Access the root node.
ii. Access the eldest child of the currently accessed node.
iii. If the currently accessed node is a parent node then go to step vii.
iv. If the next eldest twin node of the current node exist, access it and go to step iii.
v. If the stack is empty then finish.

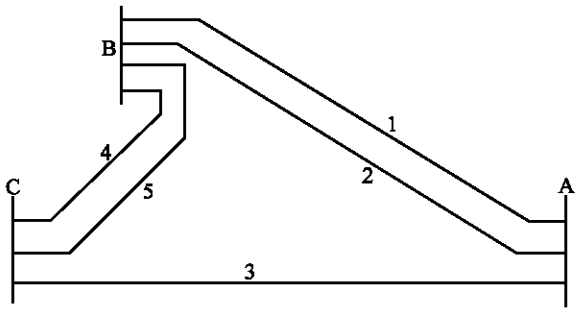

Fig. 2: Sample system-1

Table 1: Minimal cut-sets for sample system-1

Order of minimal cuts	Minimal cuts		
1	-		
2	1,2	1,3	2,3
3	3,4,5		

vi. Unstack a node pointer and if this node's next eldest twin node exists, access the next eldest twin node and go to step iii, otherwise go to step v.
vii. Stack a pointer to the current node and go to step ii.

Representation of trees in a computer system

Linked list processing: When using linked list to represent a tree structure, it is necessary for two pointers to be associated with each node of the tree. The first pointer indicates the first child of the node, i.e., the left most child, the second pointer indicates the presence of a child of the same parent, i.e., a twin pointer.

The insertion and deletion of nodes does not require the rewriting of the nodes records as is the case with sequential listing of a tree. One further advantage of the linked list structure over a sequential list is that it is possible to access a node without having to pass through every node in the tree. The steps involved in representing a network in the form of linked list are as follows:

i. Access a root node of the network.
ii. Store the root node data.
iii. Traverse the sub-tree to access the child nodes of the root node.
iv. At each child node, store the node data, set a root pointer to the child node data and set a child node pointer to the root node data.
v. Go to step i unless all the root nodes on the network have been accessed.

Application

Sample system-1: A network of five transmission lines connecting three stations is considered (Fig. 2). Stations B and C are supplied from Station A. The following two criteria for system failure are adopted.

Table 2 : Minimal cut-sets for sample system-2

Order of minimal cuts	Minimal cuts			
1	-			
2	11,12			
3	3,9,12	11,4,10		
4	3,5,7,12	3,2,1,12	3,2,5,12	3,1,7,12
	3,9,4,10	11,4,2,6	11,4,8,6	11,4,1,2
	11,4,1,8	3,2,1,4		
5	3,9,4,2,6	3,9,4,8,6	3,9,4,1,2	3,9,4,1,8
	3,1,7,4,10	3,5,7,4,10	3,2,1,4,10	3,2,5,4,10
	3,2,1,4,8	3,1,7,4,2	3,1,7,4,8	3,2,1,4,6
	3,2,5,4,6			
6	3,1,7,4,8,6	3,5,7,4,8,6	3,5,7,4,2,6	

Table 3: Comparison of the three sample systems

Order of minimal cuts	Minimal cuts			
1	-			
2	27,19	26,19	19,25	20, 27
	26,20	20,25	27,21	26,21
	25,21	27,23	26,23	25,23
3	-			
4	9,6,11,19	9,6,11,20	9,6,11,21	
	9,6,11,23	9,6,11,19	9,7,11,20	
	9,7,11,21	9,7,11,23	9,8,11,19	
	9,8,11,20	9,8,11,21	9,8,11,23	
	10,6,11,19	10,6,11,20	10,6,11,21	
	10,6,11,23	10,7,11,19	10,7,11,20	
	10,7,11,21	10,7,11,23	10,8,11,19	
	10,8,11,20	10,8,11,21	10,8,11,23	

i. The system fails if any one of the stations B and C is left without supply.
ii. The system also fails, owing to line overloading, if the combined load of B and C is carried by a single line.

The TOP event in a fault tree for any system is the failure of the entire system, which in sample system-1 is either the event that station B or station C is left without supply, or the event that the combined load of B and C is carried by a single line. The construction proceeds downward from the TOP event. The rectangles indicate subsystem failures which can be further decomposed and the circles, component failures which can not. Based on the given definition and the physical network diagram, the fault tree can be constructed (Fig. 3). The tree is terminated whenever a component failure is reached.

In this sample system, moving one level down from the TOP event, an OR-gate is passed, which indicates three cuts, {P}, {Q} and {R}. Each of these can be either decomposed into more cuts, or developed into a cut containing more failure events, according to the type of gate below it. For example, {P} is first developed into the cut {1,2,S}, S being the event that the supply from station C is interrupted, next S is decomposed into the cuts {Z} and {3}, therefore {1,2,S} is replaced by the cuts {1,2,Z} and {1,2,3} and so on. The set of cuts obtained in the successive steps are:

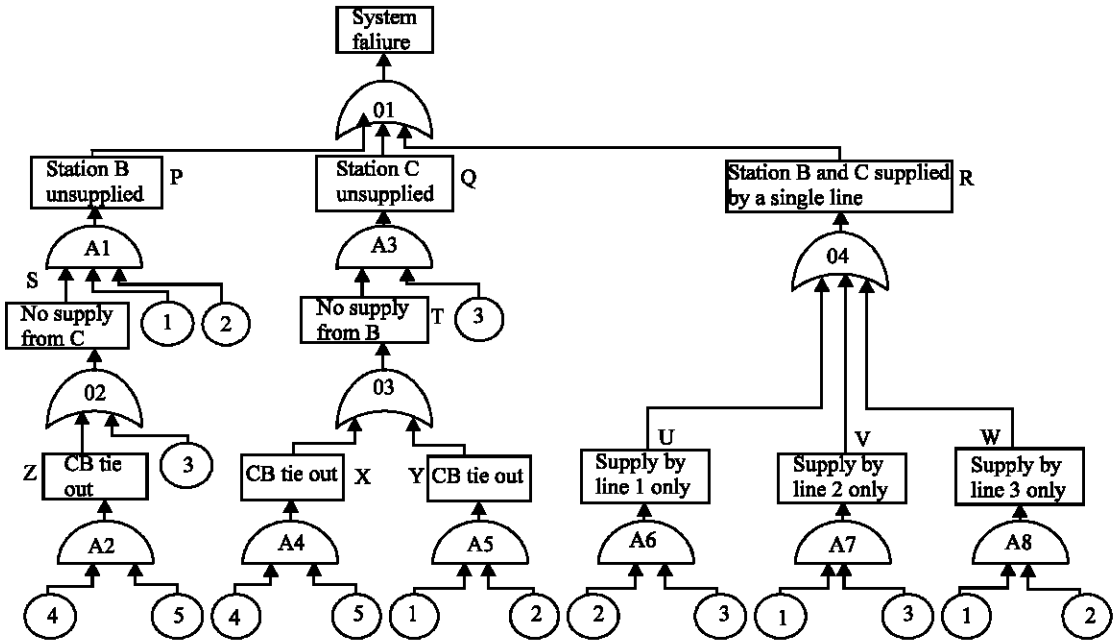

Fig. 3 : Fault tree representation of sample system-1

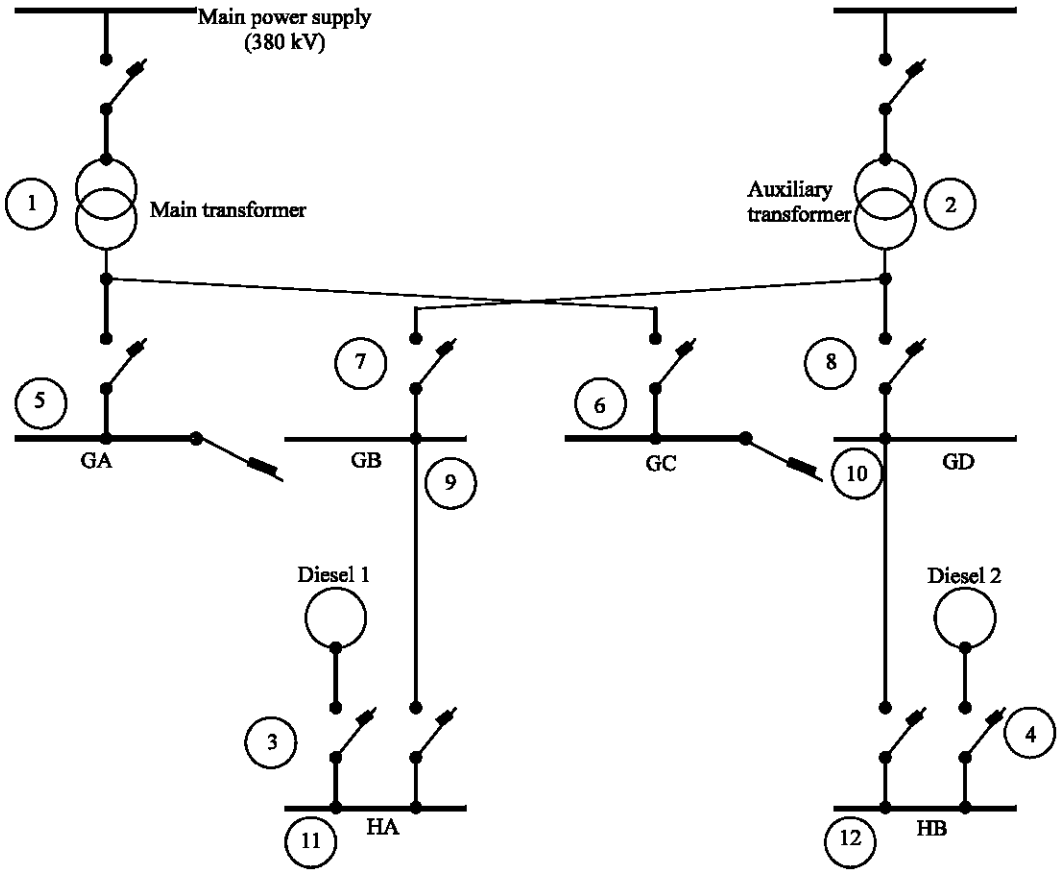

Fig. 4: Sample system-2

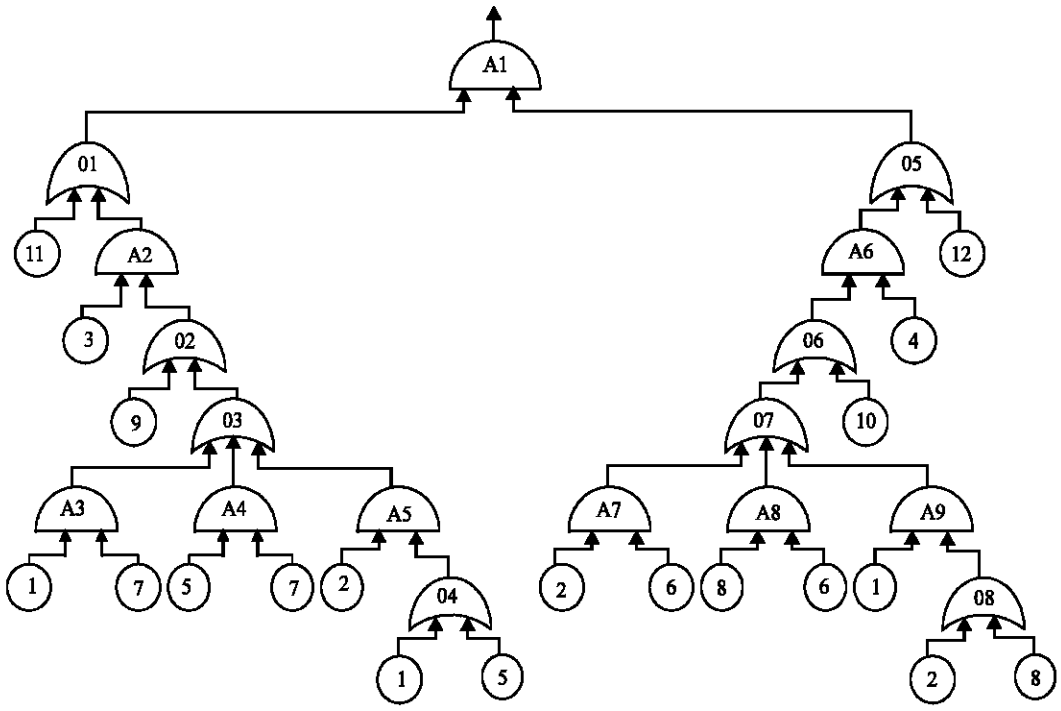

Fig. 5 : Fault tree representation for sample system-2

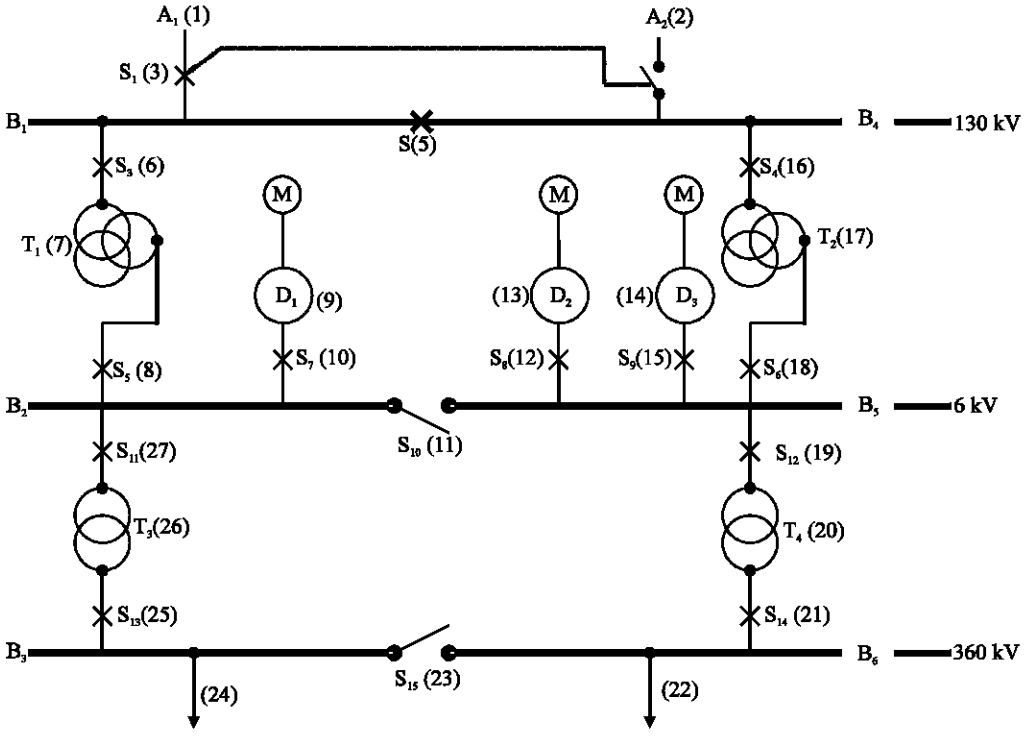

Fig. 6: Sample system-3

Table 4: Comparison of the three sample systems

System	No. of components	No. of minimal cuts	Computing time (seconds)	
			Fortran using single processor	Data structures
1	5	4	38	12
2	12	29	110	46
3	27	36 (only 2nd and 4th order cuts)	170	65

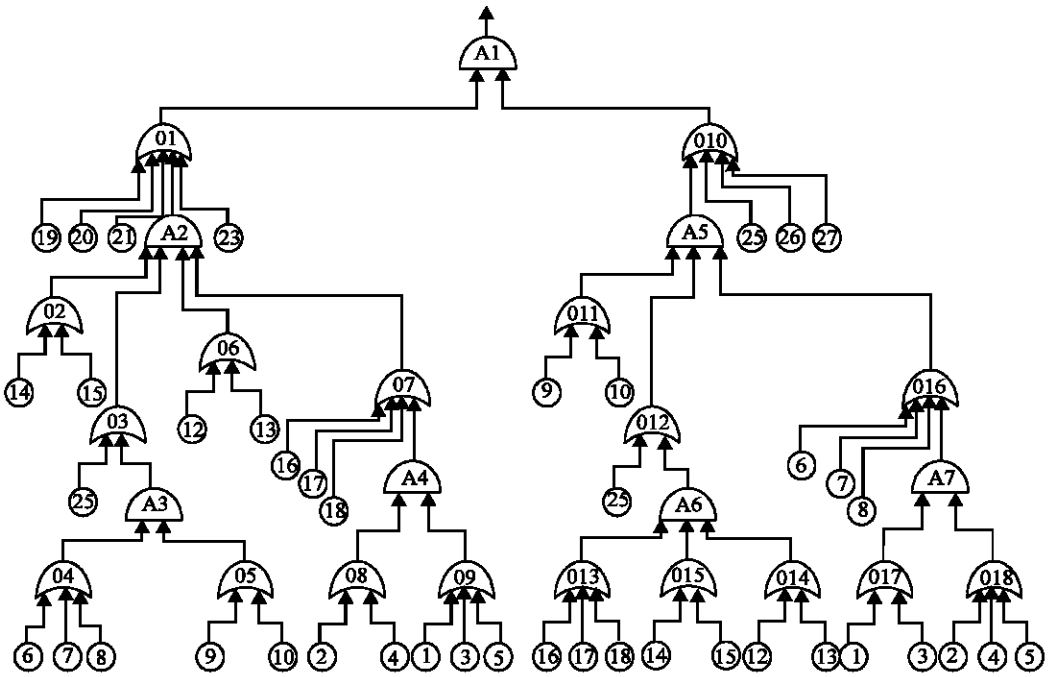

Fig. 7: Fault tree representation for sample system-3

{P}	{1,2,S}	{1,2,Z}	{1,2,4,5}
		{1,2,3}	{1,2,3}
{Q}	{3,T}	{3,X}	{3,4,5}
		{3,Y}	{3,1,2}
{R}	{U}	{2,3}	{2,3}
	{V}	{1,3}	{1,3}
	{W}	{1,2}	{1,2}

In the last step all subsystem failures are eliminated and cut-sets consists of only component failures. Some of the cuts obtained are minimal cuts and others not. For example {3,1,2} is a cut but cannot be minimal because {1,2} is also a cut. The minimal cuts are listed in Table 1.

Once all the minimal cuts are identified, the system failure probability can be calculated from the component failure data provided. In this case, the failure probability P(T) is given by, $P(T) = P_3 P_4 P_5 + P_2 P_3 + P_1 P_2 + P_1 P_3$
A C-program has been written to implement this technique.

Sample system-2: Fig. 4 shows the sample system-2. If the main supply fails, the auxiliary network and the two diesel engines (diesel 1 and diesel 2) take over. The power supply function is said to be operative if voltage is present across either of the bus-bars HA or HB.

The various components in the fault tree, numbered as 1 to 12 can be easily identified from the power supply system in Fig. 4 which are given within circles. The Fault Tree and the minimal cuts for the sample system-2 are given in Fig. 5 and Table 2 respectively.

Sample system-3: The electrical system shown in Fig. 6 is considered. The following symbols and notations are used:

A_1, A_2	-	high voltage supply lines
B_1, …, B_6	-	electrical power bars
D_1, D_2, D_3	-	no-break supply groups
S_1, …, S_{10}, S	-	circuit-breakers
T_1, …, T_4	-	power transformers
M	-	synchronous motor

The two lines A_1 and A_2 come from two different power-generating stations; A_1 is operating, while A_2 is

stand-by. Each of the two transformers T_1, and T_2 can supply the whole power to the plant. If both A_1 and A_2 are disconnected from B_1 and B_2, then B_3 and B_4 are fed by D_1, D_2, and D_3. Each of these groups is driven by a diesel engine and can supply the whole power required at B_3 and B_4. In normal conditions, the groups are permanently rotated by the three synchronous motors M. Here the chosen TOP event is 'no power to B_5'. In this example, the short circuits in the bars are not taken into account. The two switches S and S_{10} are bidirectional. The fault tree for the sample system is shown in Fig. 7. The minimal cut-sets (MCS) extracted from the fault tree is given in Table 3. The time required for estimating the overall system reliability is summarized in Table 4 for the three sample systems. The specification of the sample systems are also given in Table 4.

Since the proposed new technique uses the dynamic storage allocation, the fault trees for any complex system of any size can be represented in a computer without modifying the existing program. Only the input has to be changed.

The computation time is one of the important factors which determines the efficiency of any algorithm. Furthermore, computer time is much precious with the increasing complexity of modern power system. From Table 4, it is evident that the algorithm using data structures saves more than 50% of the computer time.

REFERENCES

1. Alain, P. and G. Michel, 1986. System reliability evaluation and prediction in engineering. North Oxford Academic Publishers Ltd.
2. Endrenyi, J., 1980. Reliability modeling in electric power systems. John Wiley and Sons Ltd.
3. Roy, B. and N.A. Ronald, 1988. Reliability assessment of large electric power systems. Kluwer Academin Publishers.
4. Allan, R.N., R. Billinton and M.F. De Oliveira, 1976. An Efficient Algorithm for Deducing the Minimal Cuts and Reliability Indices of a General Network Configurations. IEEE Trans. Reliability, R-25: 226-233.
5. Arnon, R., 1980. Decomposition Methods for Fault Tree Analysis. IEEE Trans. Reliability, R-29: 136-138.
6. Dhillon B.S. and Chanan Singh, 1981. Engineering Reliability, New Techniques and Applications. Wiley Interscience Publications.
7. Martin, F.C., 1978. Directed Graph Techniques for the Analysis of Fault Trees. IEEE Trans. Reliability, R-27: 7-15.
8. Stefan, A., 1978. Reduced State Enumeration–Another Algorithm for Reliability Evaluation. IEEE Trans. Reliability, R-27: 101-105.
9. Awoscope, C.O.A. and T.O. Akinbulire, 1990. A Fixed Minimal Path Array Method of Evaluating the Reliability Indices of Power Systems. Electric Power Systems Research, pp: 167-176.
10. Dean, B.W., S.H. Jason, R.D. Rlaph and M.R. Glenn, 1977. Fault Tree Analysis using Bit Manipulation. IEEE Trans. Reliability, R-26: 95-98.
11. Lin, P.M., B.J. Leon and T.C. Huang, 1976. A New Technique for Symbolic System Reliability Analysis. IEEE Trans. Reliability, R-25: 2-14.
12. Hideo, T, L.T. Fan, F.S. Lai and K. Toguchi, 1983. Fault Tree Analysis by Fuzzy Probability. IEEE Trans. Reliability, R-32: 453-457.
13. Brian, B., 1980. Data Structures. Blackie and Sons Ltd.
14. Chanan, S. and B. Roy, 1977. System reliability modeling and evaluation. Hutchinson and Co., Ltd.
15. Balbir, S.D. and C. Singh, 1979. On Fault Trees and other Reliability Evaluation Methods. Microelectronics and Reliability, 19: 57-63.
16. Bennetts, R.G., 1975. On the Analysis of Fault Trees. IEEE Trans. Reliability, R-24: 175-185.
17. Camarda, P., F. Corsi and A. Trentadue, 1978. An Efficient Simple Algorithm for Fault Tree Automatic Synthesis from the Reliability Graph. IEEE Trans. Reliability, R-27: 215-221.