1-1-2008

Importance analysis of fault trees by visual inspection

Gurvinder Kaur Bains
Ryerson University

IMPORTANCE ANALYSIS OF FAULT TREES BY VISUAL INSPECTION

Gurvinder kaur Bains

A project presented to Ryerson University
in partial fulfillment of the
requirements for the degree of
Master of Engineering
in the program of
Electrical and Computer Engineering

Toronto, Ontario, Canada, 2008

Author's Declaration

I hereby declare that I am the sole author of this project.
I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

Gurvinder Kaur Bains

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Gurvinder Kaur Bains

IMPORTANCE ANALYSIS OF FAULT TREES BY VISUAL INSPECTION

© Gurvinder kaur Bains

Master of Engineering

Department of Electrical and Computer Engineering

Ryerson University, 2008

Abstract

To achieve high reliability for any system, it is necessary to identify the components and the subsystems that have the greatest impact on its reliability. Such items can be identified using *importance measures* that rank the items quantitatively according to their contribution to the system unreliability. Taking into consideration the complexity and time involved in computing these measures we have proposed an algorithm that can pinpoint the most important component just by visually inspecting the fault tree. Calculations whenever required, involve simple arithmetic. It gives the user freedom from the complex calculations, save their time and performs the intended task without the use of software tools. Then the generalization of this work has been proposed that ranks the components of the fault tree. We have illustrated both the algorithms for the fault trees without as well as with repeated events.

Acknowledgements

I express my deepest gratitude to my supervisor, Dr. Olivia Das, for her patient guidance, encouragement and invaluable suggestions she has provided me throughout the time I have been her student. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. I have learned a lot from her.

I would like to thank my beloved husband Ranjit for providing me the opportunity to study further. Without his support it won't have been possible for me to pursue my graduate studies. Thanks must also go to my in-laws family for their prayers and encouragement, especially to my sister-in-law who is like an elder sister to me.

I would express a deep sense of gratitude to my parents (India), especially to my dearest mom, who has always stood by me like a pillar in times of need and to whom I owe my life for her unconditional love, encouragement, moral support and blessings. Special thanks are due to my one and only loving brother, Harman (Australia) who always strengthened my morale by standing by me in all situations regardless of the distance. I welcome the newest addition to our family, my bhabhi Navdeep.

Above all, I would like to thank god for sending such wonderful people into my life and helping me to climb one more step in the ladder of life.

Table of Contents

List of Tables

List of Figures

List of Acronyms

SA	Sensitivity Analysis
IA	Importance Analysis
FTA	Fault Tree Analysis
SI	Structural Importance
RI	Reliability Importance
BIM	Birnbaum Importance Measure
BDD	Binary Decision Diagram
MCS	Minimal Cut Sets
MTTF	Mean Time to Failure
MTFF	Mean Time to First Failure
MTTR	Mean Time to Repair
MTBF	Mean Time Between Failures

Chapter 1

Introduction

This chapter gives an introduction and overview of our project. It covers the introduction and motivation, the main contributions and the project organization.

1.1 Introduction and Motivation

In today's technological world nearly everyone depends upon the continued functioning of a wide array of complex machinery and equipment for their everyday health, safety, mobility and economic welfare. People are interested in reliable systems; they expect their cars, computers, electrical appliances, lights, televisions, etc. to function whenever they need them - day after day, year after year. When these systems fail, the results can be catastrophic: injury, loss of life and/or occurrence of costly lawsuits. More often, repeated failure leads to annoyance, inconvenience and a lasting customer dissatisfaction that can play havoc with the responsible company's marketplace position.

Reliability engineers are often called upon to make decisions as to whether to improve a certain component or components in order to achieve minimum required system reliability. Fault avoidance and fault tolerance are the two approaches to improve the reliability of a system. Fault avoidance is achieved by using high-quality and high-reliability components, and is usually less expensive than fault tolerance. Fault tolerance, on the other hand, is achieved by redundancy. Redundancy can result in increased design complexity and increased costs through additional weight, space etc.

Before deciding whether to improve the reliability of a system by fault tolerance or avoidance, a reliability assessment for each component in the system should be made.

Once the reliability values for the components have been quantified, an analysis can be performed in order to determine if system's *reliability goal is met, which component or module is contributing the most to the unreliability, what is the most cost effective way to improve the reliability?* These questions require sensitivity analysis (SA) of the reliability results.

Sensitivity Analysis (*Importance Analysis*) also referred as what-if analysis is simply the study of the relationship or correlation between outputs of a model to its inputs. It is defined as the partial derivative of the measure with respect to input parameters. SA can prove very useful in understanding the behavior of a system. SA of system reliability can be used to guide system optimization, identify the parts of system model sensitive to error and find system reliability bottlenecks [28].

Several reliability models have been developed to evaluate the reliability of software systems. SA can determine the most sensitive parameter and most sensitive component for such models. Jung et al [5] studied the sensitivity analysis of the reliability of component-based applications. They showed that SA can not only determine the parameters and the components affecting the reliability of the system but can also find the most sensitive interaction between the components and the most sensitive relative error component. Gokhale [8] performed sensitivity analysis based upon the architecture of the software.

SA analysis has also been defined in context of hardware reliability models (most notably traditional fault tree analysis) as a way to assess the relative importance of a component to the reliability of a system. Several measures have been defined to assess the component importance. These *importance measures* [23, 31, 32] rank the items quantitatively according to their contribution to system unreliability. They are used to detect design weaknesses and component failures that are critical to the proper functioning of a system. Overall, these measures assist in identifying the components whose improvement is most likely to yield the greatest improvement in system reliability.

The importance [17, 23, 31, 32] measures are categorized into two types: *Structural-Importance* (SI) and *Reliability Importance* (RI). In SI measures, the importance of a component to the system is assessed by the virtue of its position in the system without considering the reliability of the component. These measures can be used even if the component reliability is unknown. On the other hand, the RI measures consider both the position of the component and the component reliability, thus generally provide more information for generating the ranked list than the SI measures interpretations. But the probabilistic information required for their calculations might not be available in practice and computations involved in quantifying these measures can become extensive for large complex systems. In such situation SI measures must be used.

The currently applied SA procedure is based on three main steps:

- Ranking of components according to their importance to system failure.
- Definition of an improved design alternative.
- Assessment of the effects of the adopted design solution to system failure probability.

Step 1. Component Ranking

One of the most useful results in system analysis is the component importance index, which represents a measure of the relative contribution to system failure probability, i.e. the occurrence of a given system failure mode, due to component failure. Thus, components can be ranked according to their importance in the system. Consequently, the weakest system points, to be considered for subsequent design improvement, can easily be identified on a rational and objective basis: they are made up of components whose failure modes present the highest importance index.

Step 2. Design Modification

Having identified the weakest system points, the design can be improved by identifying and implementing more reliable solutions. Following the design modification the system model is updated.

Step 3. Re-analysis of the Model

The modified model is then re-analyzed to assess the improvements made.

In the above three steps, component ranking is the most critical. Ranking can be performed by either of the two types of measures (RI, SI). But both have their own drawbacks. SI measures do no take into account the reliability of component, and thus they can not distinguish between components that occupy the similar structural positions but have drastically different reliabilities. On the other hand RI measures are more accurate; however, their computations involve complex calculations.

Since the computation of these measures involves complex calculations such as partial derivates, they are computed with the help of software tools. This motivated us to find a simpler way to identify the most significant component(s), as for *improving the reliability* of the *sy*stem the engineers are generally interested in finding the most "sensitive" or "important" component to the system reliability. The improvement in this component will lead to the maximum improvement in the system reliability. As a result we developed an algorithm that performs the intended task with no or simple arithmetic calculations, thus eliminating the need of any software tool.

1.2 Contributions

For sensitivity analysis of a fault tree model we have developed two algorithms. The first is developed keeping in mind that to improve the reliability of the system we need to find the component which affects the reliability of the system the most. The second is developed to rank all the components of the system according to their importance in the system reliability as sometimes the engineers may need to concentrate on more than one component as these components may also be playing a significant role in the system unreliability.

The principal contributions of this project are as follows:

- An *algorithm* to find the *most sensitive component* of a fault tree has been proposed. This algorithm works for fault trees with non-repeated as well as

repeated events. Unlike the previous work done in this field, this algorithm requires no or simple calculations. The user can find the most "sensitive" component not only without using any software tool but just by looking at the tree structure. The algorithm is shown to work accurately by comparing its results with Birnbaum importance measure.

- *Ranking algorithm* has been developed which not only accurately finds the most sensitive component for all the tree structures but also precisely *ranks* the components of the system for most of them. The *Ranking* is also shown to work by comparing it with the ranking of Birnbaum importance measure.

1.3 Project Organization

Chapter 2 provides the background needed for our project. It covers the basic terminology related to reliability, the various reliability analysis models, fault trees and the sensitivity analysis of fault trees.

Chapter 3 contains the proposed algorithm to find the most sensitive component of the fault tree. It has been shown to work for the fault trees with repeated as well non-repeated events. The results are compared with the Birnbaum importance measure computed using the demo version of software, *Aralia* [39].

Chapter 4 contains the generalized algorithm, for ranking the basic events of the fault trees according to their importance to the system reliability. This algorithm is illustrated with the help of different fault tree structures. The results are compared with the Birnbaum importance measure computed by Aralia WorkShop.

Chapter 5 includes the summary, conclusion, advantages and limitations of our work.

Chapter 2

Background

This chapter introduces the preliminary concepts that form the foundation of our work. In Section 2.1 reliability is defined. Section 2.2 deals with the reliability metrics. In section 2.3 different reliability models are discussed. Fault tree, the model used in our project is explained in section 2.4. Fault tree Analysis methods are covered in section 2.5. Sensitivity analysis of fault tree analysis is addressed in section 2.6.

2.1 Reliability

Reliability [40] is a broad term that focuses on the ability of a product to perform its intended function. The product could be an electronic or mechanical hardware product, a software product, a manufacturing process, or even a service. Mathematically speaking, assuming that an item is performing its intended function at time equals zero, reliability can be defined as the probability that an item will continue to perform its intended function without failure for a specified period of time under stated conditions. Reliability, *R(t)* of a system *S* can be expressed as:

$$R(t) = Pr(\text{S is fully functioning in } [0, t])$$

System reliability, by definition, includes both parts of the system: hardware and software. Different metrics are used to measure software and hardware reliability. Moreover, due to different failure processes, separate reliability models had been defined by various authors. Number of software reliability models had been proposed in [27, 28, 29, 30, 33]. But we are concerned only with hardware reliability.

2.2 Reliability metrics

The metrics [41] which are used for assessing the reliability are:

- **Failure rate**: The expected rate of occurrence of failure or the number of failures in a specified time period. Failure rate is typically expressed in failures per million or billion hours.
- **MTTF** (Mean Time to Failure): Average time it takes for a system to fail.
- **MTBF** (Mean Time between Failures): Number of hours that passes between failures. It is expressed in hours.
- **Reliability** can be defined as the probability that the component or system remains operating from time zero to time t, given that it was operating at time zero. It answers the question: "How likely is it that my system will remain operational over a period of time?" Because reliability is expressed as a probability, it is always a value between 0 and 1.
- **Availability** of a component or system is defined as the probability that the component or system is operating at time t, given that it was operating at time zero. It answers the question: "How likely is it that my system is operating at X hours?"
- **Unreliability** is the compliment of reliability. It indicates the likelihood that a system *cannot* continuously operate up to a specified point in time.
- **Unavailability** of a component or system is defined as the probability that the component or system is not operating at time t, given that is was operating at time zero.

2.3. Reliability Modeling

System reliability analysis refers to the evaluation of the reliability of a system based on the reliabilities of its elements. It also includes computation of other measures such as failure (hazard) rate, failure (occurrence) frequency, MTTFF (Mean Time to First Failure), MTTF (Mean Time to Failure), MTTR (Mean Time to Repair), and MTBF (Mean Time Between Failures).

There are many options [18] to predict the reliability of the system including, making an educated guess based on experience with similar systems, using *discrete-event simulation* such as Monte-Carlo simulation to model the system or *construct analytical models* of the system. Discrete event simulation is a program that mimics the dynamic behavior of the modeled system and provides measures of the system behavior. The analytical approach involves the determination of a mathematical expression which describes the reliability of the system, expressed in terms of the reliabilities of its components. These models, which are an abstraction of the system, might not predict the system behavior that well but under certain circumstances they can provide information which cannot be obtained by any other methods.

There are a wide range of analytical models for the system designer to choose. Each model has its strength and weakness in terms of accessibility, ease of construction, efficiency, accuracy of solution algorithms, and availability of software tools. Reliability modeling approaches can further be divided into two categories: *non state space analytical modeling* and *state space analytical modeling* [12].

Non state space models: Models [18] like reliability block diagram, fault trees, reliability graphs can be easily formulated and solved for system reliability, system availability and system MTTF. Each component can have attached to it, probability failure; failure rate; distribution of time to failure; steady state or instantaneous unavailability. The two main assumptions used by these models are statistically independent failures and independent repair units for the components. The limitation of these models is that they can not represent dependencies occurring in the real time systems [29, 30] such as imperfect coverage, correlated failure, repair dependencies, transient and intermittent faults, standby systems with warm spares, and so forth. The state space models which are discussed next, overcome this limitation of non-state space models.

State space Models: Models such as Markov models [1, 9, 14, 21, 22] are capable of representing important system behavior. The major *drawback* of Markov method is that it

is not always intuitive and markov diagrams for large systems are generally exceedingly large, complicated and difficult to construct. However, Markov models may be used to analyze smaller systems with strong dependencies requiring accurate evaluation.

Monte-Carlo simulation-It is difficult to find an exact analytical expression or algorithm for every scenario. It is particularly difficult when complex dependencies exist. Such dependencies may include warm standby components, shared repair resources, repair actions based on the state of the system and so forth. To overcome the difficulties in analyzing systems with complex dependencies, *Monte Carlo simulation* [26] can be used to calculate system reliability by simulating the failure of components at times distributed according to their failure rates. The *disadvantage* of this approach is that if the number of simulations performed is not large enough, this method can be error prone. In addition, performing a large number of simulations can be extremely time-consuming and if minor changes occur the simulation must be rerun at a considerable cost.

2.4 Fault Trees

Among all the reliability models, Fault Trees [20] have been most widely used as they provide a compact, graphical, intuitive method to analyze system reliability. The Fault tree is a pictorial representation of the combination of *events* that can cause the occurrence of an undesirable (system failure) *event*. By means of *logic gates* an *event* at level *i* is reduced to a combination of lower-level *events*. This process is iterated until the *basic events* (that cannot be reduced further) are reached. The occurrence of each *event* is denoted by logic 1 at that node; otherwise the logic value of the node is 0.

Each *gate* has inputs and outputs. The input to a *gate* is either a *basic event* or output of another *gate*. The two most commonly used *gates* are AND and OR *gate*. The output of an AND *gate* is a logic 1 if and only if all its inputs are logic 1. The output of an OR *gate* is logic 1 if and only if one or more inputs are logic 1. There is a single output called the *top event* representing system failure. The recent developments in the modeling features of fault trees include the concept of dynamic fault trees [1], which add the sequential

notion to the traditional fault tree approach. Hence, system failures can then depend on component failure order as well as combination [15].

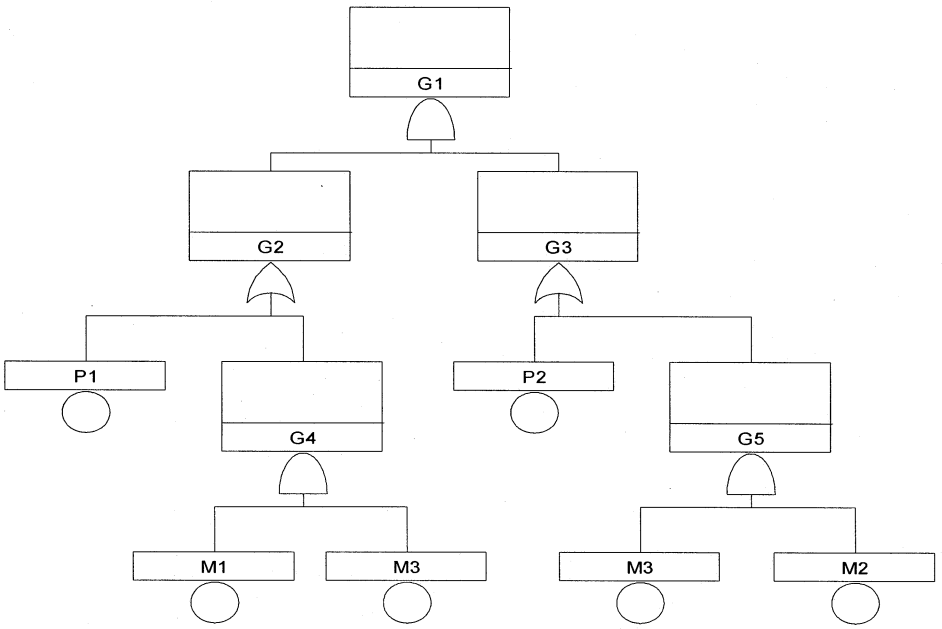

Figure 2.1: A fault tree with repeated event M3

Consider a fault-tolerant computer system with two processors and three memory modules. Assuming that one of the memory modules M3 is shared, M1 is private to processor P1 and M2 is private to processor P2; the system will operate as long as there is at least one operational processor with access to either a private or shared memory. This system can be modeled by a fault-tree as shown in Figure 2.1. Module M3 is representing a repeated event in this fault tree.

2.5 Fault tree Analysis

Analysis of fault tree begins with an enumeration of *minimal cut sets* i.e. the smallest combinations of component failures which, if they occur, will cause the top event to occur. A *cut set* is combination of primary effects sufficient for top event. Once the minimal cut sets are obtained, probability analysis can be performed to obtain the top event probability and quantitative importance of components. But the number of minimum cut sets usually increases with the size of trees and this leads to a complexity in

the evaluation of a fault tree. Both static and dynamic trees can be analyzed by using the Sum-Of-Disjoint-Products algorithm [19, 25, 31, 34]. The use of Binary Decision Diagrams (BDDs) [3, 7] in fault tree analysis provides both an accurate and efficient means of analyzing a system. This method does not analyze the fault tree directly, but converts the tree to a binary decision diagram, which represents the Boolean equation for the top event. The difficulty, however, lies with the conversion of the tree to the BDD. For Complex fault trees, a Modular approach [15] has been proposed for the efficient analysis of both static and dynamic fault trees. It provides a combination of BDD solution for static fault trees and Markov chain solution for dynamic fault trees. For dynamic fault tree, the occurrence of the top event depends on not the combination of basic events, but also the occurrence order of the basic events. Cut sequence is a set of basic events that fall in a particular order. Dong [2] uses cut sequence set for analysis of dynamic fault trees.

2.6 Reliability Importance Analysis

Importance analysis (Sensitivity Analysis) has been defined in context to Fault trees as a way to assess the relative importance of a component to the reliability of the system. Several importance measures have been defined to assess component importance [24] in order to capture the combined effect of structural and probabilistic contributions to the system reliability from a basic component. These various measures have generally fallen out of favor [24] as they counterintuitive or inconsistent results. However three measures which remained most popular are discussed below.

Birnbaum importance measure: Birnbaum [38] measure determines the maximum increase in risk when component is failed compared to when component is operating. It is defined as a partial derivative of system reliability with respect to individual component failure rate. This method can be considered as a form of *sensitivity analysis*, as the index gives an indication of how system reliability will change with changes in component reliability. While the *Birnbaum importance measure* is useful, it does not directly consider how likely event is to occur. This measure is independent of the actual unavailability of event, which can lead to assigning high importance measures to events

that are very unlikely to occur and may be very difficult to improve. Another *issue* regarding this index is that it can not be used to predict several changes at the same time, i.e. reliability changes in several components at a time [12]. However, the index can be used to determine effects of changes, which is not possible for all indices.

Many new measures have been defined taking this measure as a base. Lee [16] extended the Birnbaum importance measure to a subsystem level (gate-event level in a fault tree). Beeson [6] developed an extension of Birnbaum measure for the analysis of noncoherent systems. Butler [35] formulated an importance ranking among the components based upon minimal cut sets of the system and showed that the so called cut-importance ranking is consistent with the ranking induced by Birnbaum reliability importance measures when component reliabilities are equal or close to 1. Meng [17] proposed a new method to compute Birnbaum importance measure in terms of minimal cut sets and minimal path sets. Yong [10] proposed a modular approach to sensitivity analysis in which they perform sensitivity analysis of static modules using BDD and sensitivity analysis of dynamic modules using markov chains. And then they combine the sensitivity results into system-level sensitivity and calculate the measure of importance with respect to different parameters such as failure rate and component type.

Criticality importance measure: Given that the top event occurs, it determines the probability that the failure is a result of the failure of the component. While the Birnbaum importance measure considers only the conditional probability that event is critical, the *Criticality importance measure* [13] also considers the overall probability of the top event occurrence due to event. It modifies the Birnbaum importance measure by adjusting for the relative probability of basic event to reflect how likely the event is to occur and how feasible it is to improve the event. These modifications enable the Criticality importance measure to focus on truly important basic events and make it possible to compare basic events between fault trees. Wang [4] modifies the Component criticality importance measure to find *Failure Criticality Index* for ranking the components in complex system.

Fussell-Vesely importance measure: Fussell [37] introduced another importance measure. Given that the system has failed, this measure determines the probability that

component has contributed to the system failure. It is the ratio of the probability of occurrence of any cut set containing event and the probability of the top event. Therefore, *Fussell-Vesely importance measures* are calculated quite differently than Birnbaum or Criticality importance measures. It constructed using minimal cut sets. A drawback of Fussell-Vesely's index is that it does not take into account the component's contribution to system success.

There are some more authors whose work has also gained recognition in this field. Barlow & Proschan [36] suggested that the most important component is that having the highest probability of finally causing system failure by its own failure. Natvig [32] has developed a theory supporting another measure. Here the component whose failure contributes most to reducing the expected remaining lifetime of the system is the most important one. Carot [11] proposed a new method for the reliability importance of the components by studying how the system life improves when the *mean life* of a component is improved. With this knowledge one can highlight which component (or components) must be given greater attention, when all the components are independent of each other.

The biggest issue with these importance measures is the complexity involved in their computation. The calculations involved are hard and time consuming when performed manually. Therefore software tools are generally required for their computation measures. We have proposed a new method to overcome the limitations of these measures. This method is discussed in the following chapters of the report.

Chapter 3

Most Sensitive Component

In this chapter we will discuss the algorithm proposed to find the most sensitive component of the fault tree. It works for the fault trees without as well as with repeated events. Section 3.1 contains the overview of the algorithm. Section 3.2 covers the basic assumptions of the algorithm. In section 3.3 are some basic notations used in the algorithm. The proposed algorithm is elaborated in Section 3.4. Section 3.5 deals with some special cases of the algorithm. In Section 3.5 algorithm is shown to work for fault trees with repeated as well as non repeated events.

3.1 Overview

Our main aim to develop this algorithm was to find the most sensitive component of the fault tree by just visually inspecting it. It involves no or simple arithmetic calculations. Following *bottom-up* approach, we compute two things for each event (basic and gate): *value* and *InputList.* We start the computation from the basic events and go up in the fault tree level by level. Level 0 are the basic events, level 1 gate is that whose inputs are basic events & so on. The top gate has the highest level.

For the basic event *value* will be its failure probability and the *InputList* is NULL. For a gate, *value* is computed using the values of the inputs (i.e. the children) & the type of predecessor (parent) gate and *InputList* is created from its inputs. These two things (*value & InputList*) are stored in each event in the form of an entry (*InputID, value, type, InputList*). *InputID* is the identifier of each event. The *type* indicates the type of input; it can be a basic event (BE) or a gate (OR, AND).

For the fault tree with the repeated events the *InputList* of the top gate is checked and action is taken according to its *type*.

The most sensitive component of the fault tree can be found from the *InputList* of the top gate.

3.2 Assumptions

We developed the algorithm by making the following assumptions:

1. There are 2 states of each basic event: it occurs or it does not occur.
2. Occurrences of basic events are mutually s-independent.
3. Gates are either AND or OR.
4. Failure probabilities of all the basic events are known.

3.3 Notations

e_i	basic event i
G_i	gate event i
E	$\{e_1,\ldots, e_m\}$: set of e_i
G	$\{G_1 \ldots G_n\}$: set of G_i
G_1	top gate
$G_{j,i}$	input j to gate G_i
parent(G_i)	predecessor of G_i
q(e_i)	failure probability of e_i
t(G_i)	*type* of gate G_i .It can have values {AND, OR}
t($G_{j,i}$)	*type* of input j to gate G_i . It can have values {BE, OR, AND}
v(G_i)	*value* of gate G_i
List(G_i)	InputList of gate G_i
v($G_{j,i}$)	value of the input j of gate G_i

3.4 Algorithm

Perform step 1 and Step 2 for all the gates, except step 2 for gate G1.Steps 3 and 4 performed at the top gate only.

Step 1. Create an InputList for each gate.

(a) Check $t(G_i)$ and $t(G_{j,i})$, make the entries in the $List(G_i)$ according to the following table. Each entry is of type: *(InputID, value, type, InputList).*

$t(G_i)$	$t(G_{j,i})$	ACTION
AND	BE	An entry will be made to $List(G_i)$ as
OR	BE	(e_i , $q(e_i)$, BE, Null)
AND	OR	An entry will be made to $List(G_i)$ as
OR	AND	($G_{j,i}$, $v(G_{j,i})$, $t(G_{j,i})$, $List(G_{j,i})$)
AND	AND	$List(G_{j,i})$ is added to $List(G_i)$
OR	OR	

Table 1: Rules to create the InputList of each gate

(b) Arranging the entries in the InputList

$t(G_i)$	ORDER
AND	$List(G_i)$ is arranged in the ASCENDING order of *value* of its entries.
OR	$List(G_i)$ is arranged in the DESCENDING order of *value* of its entries.

Table 2: Rules to arrange the InputList of each gate

Step 2. Find the gate value

$t(G_i)$	$parent(G_i)$	$v(G_i)$
AND	AND	*Value* of the top entry in
OR	OR	
AND	OR	The product of *value* of the entries in the $List(G_i)$..
OR	AND	The sum of *value* of the entries in $List(G_i)$.

Table 3: Rules to find the Gate value of each gate

Step 3. Check for the repeated events.

Examine List(G_1), if there are no repeated events, go to step 4.

$t(G_1)$	ACTION
OR	If a basic event in an OR gate *InputList* is repeated in an AND gate *InputList*, then all other entries of that AND gate (except this repeated event and the ones which are repeated in *InputList* of another OR gate) will be considered to have Zero importance.
	If the basic event is repeated in other OR gates, it won't affect its importance.
AND	If a basic event in an AND gate *InputList* is repeated in an OR gate *InputList*, then all other entries of that OR gate (except this repeated event and the ones which are repeated in *InputList* of another AND gate) will be considered to have Zero importance.
	If the basic event is repeated in other AND gates, it won't affect its importance.

Table 4: Rules for a fault tree with repeated events

4. Find the most sensitive component

We look for the *type* of top most entry of the List(G_1). If it is of *type* BE, then that event will be the most sensitive component. If the following entries are of type BE and have same failure probabilities then these events will also have the same importance. But if the topmost entry of List(G_1) is a gate then the topmost entry of its *InputList* will be considered. The process is continued until a basic event is hit.

3.5 Special Cases

The above algorithm is simplified for special cases of fault trees. In these cases we can pinpoint the most sensitive component just by looking at the fault tree structure.

3.5.1: A Fault tree with all OR gates

The basic event with the highest failure probability will be the most sensitive.

In Figure 3.1, all the gates G1, G2, G3 and G4 are OR gates. Therefore, among all the basic events, the most sensitive component will be the one with highest failure probability. For example, if we consider the failure probabilities of E001, E002, E003, E004, E005, E006, E007, E008 and E009 as .003, .007, .006, .001, .004, .008, .002, .005 and .003 respectively. The most sensitive component will be **E006** which has highest failure probability i.e. .008.

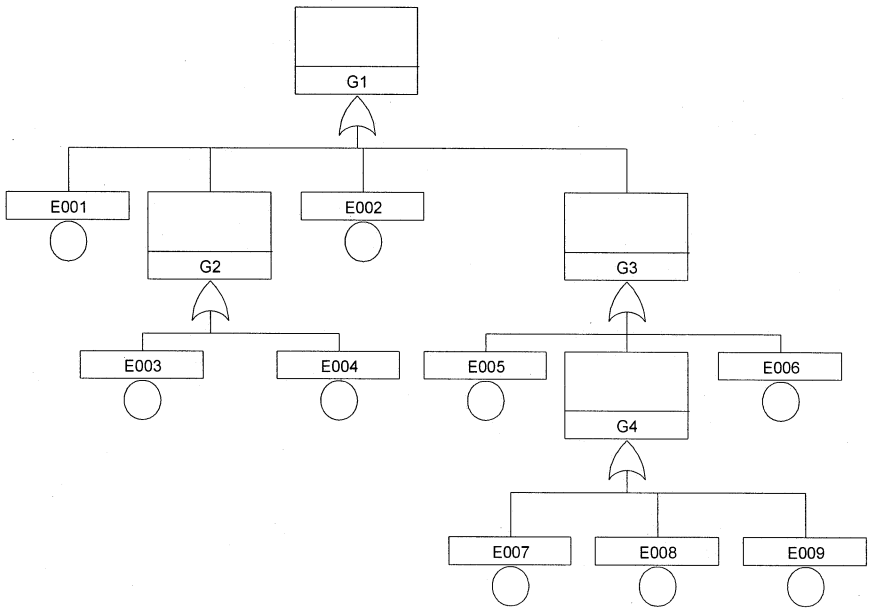

Figure 3.1: A fault tree with all OR gates

Table 5 shows the Birnbaum importance measure obtained by using *Aralia.* We can see from the table that E006 (highlighted) has the highest Birnbaum index. Therefore it is the most sensitive component. The result thus matches with the outcome of our algorithm.

Basic Event	Failure Probability	BIM
E006	**0.008**	**9.69E-01**
E002	0.007	9.68E-01
E003	0.006	9.67E-01
E008	0.005	9.66E-01
E005	0.004	9.66E-01
E001	0.003	9.65E-01
E009	0.003	9.65E-01
E007	0.002	9.64E-01
E004	0.001	9.63E-01

Table 5: Birnbaum Importance measure for a fault tree with all OR gates

3.5.2: A fault tree with all AND gates

The basic event with the lowest failure probability will be the most sensitive event of the tree.

For Figure 3.2, we considered the same failure probabilities as considered for Figure 3.1. In this case, the most sensitive component will be **E004** which has lowest failure probability i.e. .001.

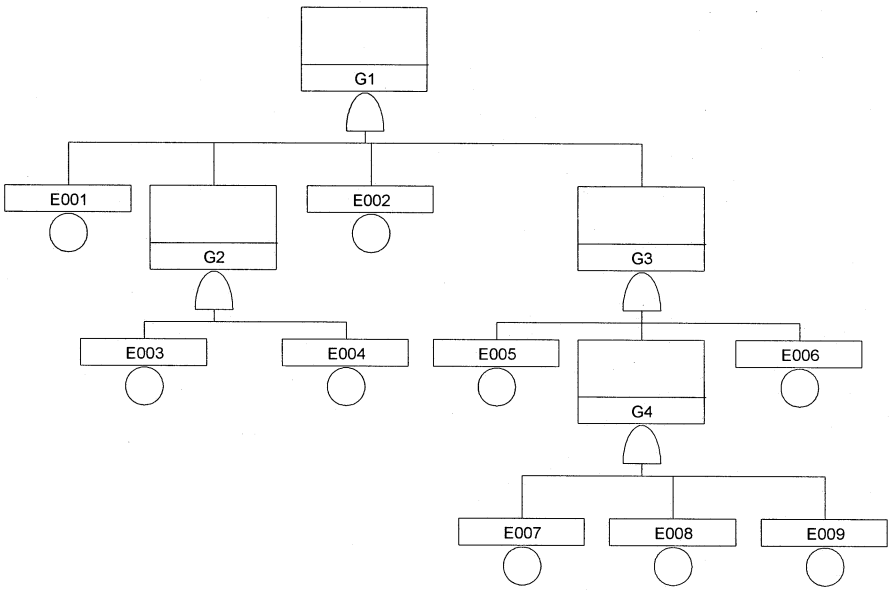

Figure 3.2: A fault tree with all AND gates

Below is the Birnbaum importance measure of the basic events of the above fault tree. It has been obtained using Aralia. Table 6 shows that **E004** is the most important component since it has the highest BIM.

Basic event	Failure probability	BIM
E004	**0.001**	**1.21E-19**
E007	0.002	6.05E-20
E001	0.003	4.03E-20
E009	0.003	4.03E-20
E005	0.004	3.02E-20
E008	0.005	2.42E-20
E003	0.006	2.02E-20
E002	0.007	1.73E-20
E006	0.008	1.51E-20

Table 6: Birnbaum Importance measure for the fault tree with all AND gates

3.5.3: A fault tree with top OR gate with basic events, AND gates as its inputs

Among the basic events which are inputs to the top gate, the one with the highest failure probability will be the most sensitive component.

In Figure 3.3, top gate is an OR gate having basic events (E001, E002) and AND gates (G2, G3) as its inputs. Among the basic events E001 and E002, the one with the highest failure probability will be the most sensitive component. No need to consider the AND gates.

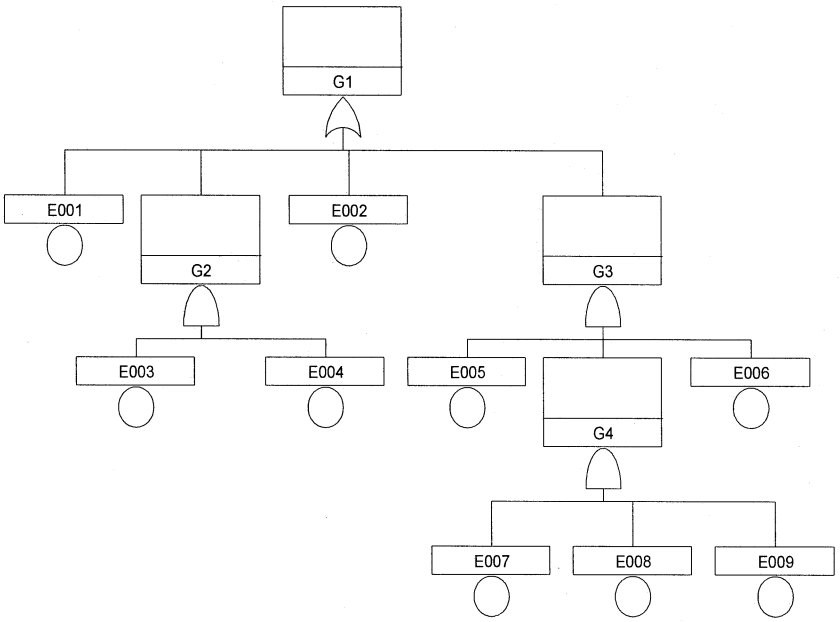

Figure 3.3: A fault tree with top OR gate with basic events, AND gates as its inputs

Considering the same failure probabilities as in the above two cases, E002 has highest probability among the basic events which are inputs to the top gate. Therefore the most significant component of this fault tree is **E002.** In Table 7 is the BIM obtained from Aralia. We can see that E002 has the highest BIM, so it is the most sensitive event according to this measure also.

Basic Event	Failure probability	BIM
E002	**0.007**	**9.97E-01**
E001	0.003	9.93E-01
E004	0.001	5.94E-03
E003	0.006	9.90E-04

E007	0.002	4.75E-10
E009	0.003	3.17E-10
E005	0.004	2.38E-10
E008	0.005	1.90E-10
E006	0.008	1.19E-10

Table 7: BIM for a fault tree with top OR gate with basic events, AND gates as its inputs

3.5.4: A fault tree with top OR gate with basic events, OR gates, AND gates as its inputs

The basic events which do not have any AND gate above them will only be considered and the event with the highest failure probability among them will be the most sensitive.

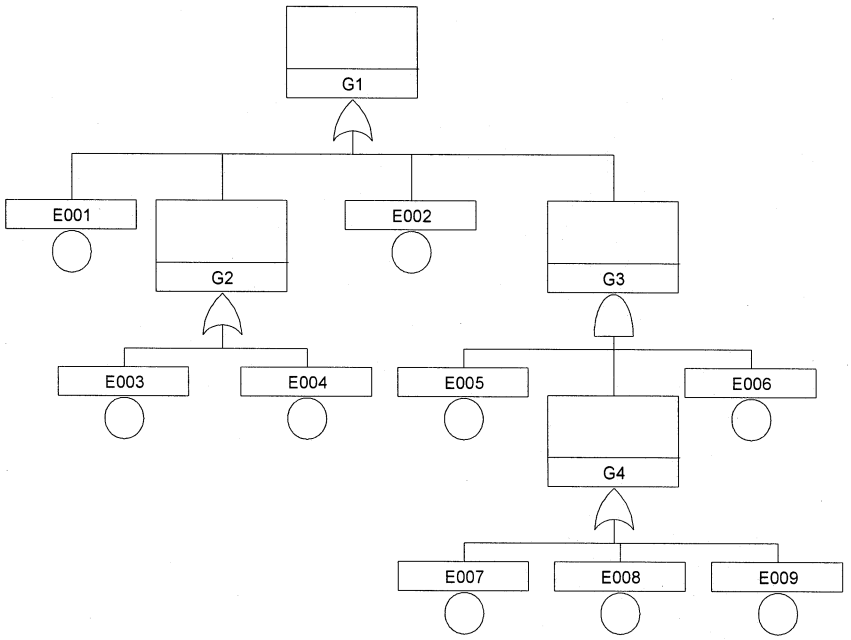

Figure 3.4: A fault tree with top OR gate with basic events, OR gates, AND gates as its inputs

In Figure 3.4, basic events E005, E006, E007, E008 and E009 have an AND gate G3 above them. These events will not be considered. Now among the basic events E001, E002, E003 and E004, the basic event with the highest failure probability will be the most sensitive. Considering the same failure probabilities event as in the above discussed cases, **E002** will be the most important component. Table 8 shows the BIM computed for this fault tree using Aralia, it also shows that **E002** is the most important component.

Basic Event	Failure Probability	BIM
E002	**0.007**	**9.90E-01**
E003	0.006	9.89E-01
E001	0.003	9.86E-01
E004	0.001	9.84E-01
E005	0.004	7.84E-05
E006	0.008	3.92E-05
E008	0.005	3.13E-05
E009	0.003	3.12E-05
E007	0.002	3.12E-05

Table 8: BIM for a fault tree with top OR gate with basic events, OR gates, AND gates as its inputs

3.6 Examples

In this section, we will consider general cases. First we consider an example of a fault tree without repeated events and then a fault tree with repeated events. The results of these examples are also compared with the Birnbaum importance measure calculated using Aralia.

3.6.1: A fault tree with non repeated Events

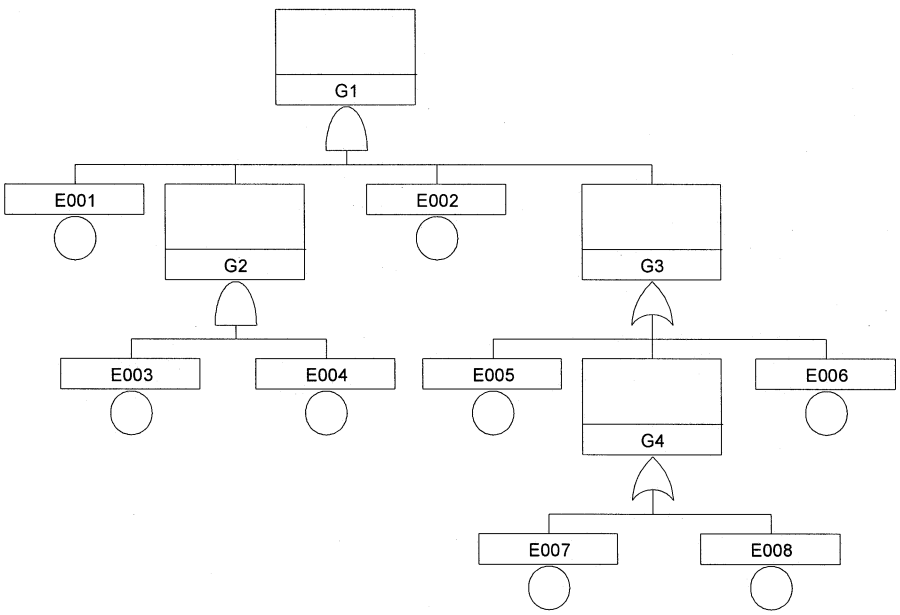

Figure 3.5: A fault tree with non repeated events

The fault tree in Figure 3.5 has basic events E001, E002, E003, E004, E005, E006, E007 and E008 with the failure probabilities .001,.004,.003,.007,.002,.006,.004 and .003 respectively Following the bottom up approach, steps 1-2 will be carried out for all the gates of the tree, except step 2 for the top gate. Step 3 is skipped because there are no repeated events. Step 4 is performed only for the top gate. The steps are shown in the tabular (Table 9) form for simplification

Gate	Step 1- Create InputList	Step 2- Gate Value
G4	t(G4) =OR, t(E007) =BE, t(E008) =BE List(G4)(E007,.004,BE,NULL) (E008,.003,BE,NULL)	t(G4) = OR, t(G3)= OR v(G4) = .004
G2	t(G2) =AND, t(E003) =BE, t(E004) =BE List(G2)(E003,.003,BE,NULL) (E004,.007,BE,NULL)	t(G2) = AND t(G1)= AND v(G2) =.003
G3	t(G4) =OR, t(E005) =BE, t(E006) = BE List(G3)(E006,.006,BE ,NULL) (E007,.004,BE ,NULL) (E008,.003,BE,NULL) (E005,.002,BE,NULL)	t(G3) = OR t(G1)= AND v(G3) = .015
G1	t(G1)=AND, t(E001)=BE, t(E002)=BE ,t(G2)=AND, t(G3)=OR List(G1)**(E001,.001,BE,NULL)** (E003,.003,BE,NULL) (E002,.004,BE,NULL) (E004,.007,BE,NULL) (G3,.015,OR,(E006,.006,BE ,NULL) (E007,.004,BE ,NULL) (E008,.003,BE,NULL) (E005,.002,BE,NULL))	

Table 9: Algorithm steps for a fault tree with non repeated events in Figure 3.5

Step 4: Most sensitive event is the top most entry of List (G1).i.e. E001

Table 10 shows the Birnbaum measure for Figure. 3.5 computed by Aralia. This result also shows that the most significant component is E001 as it has the highest BIM. Hence it has found the same component to be most sensitive as found by our algorithm.

Basic Event	Failure probability	BIM
E001	**.001**	**1.25E-09**
E003	.003	4.18E-10
E002	.004	3.13E-10
E004	.007	1.79E-10
E006	.006	8.32E-11
E007	.004	8.31E-11
E008	.003	8.30E-11
E005	.002	8.29E-11

Table 10: BIM measure for the fault tree in Figure 3.5

Example 2: A fault tree with repeated Events

Now we consider an example of a fault tree with repeated events.

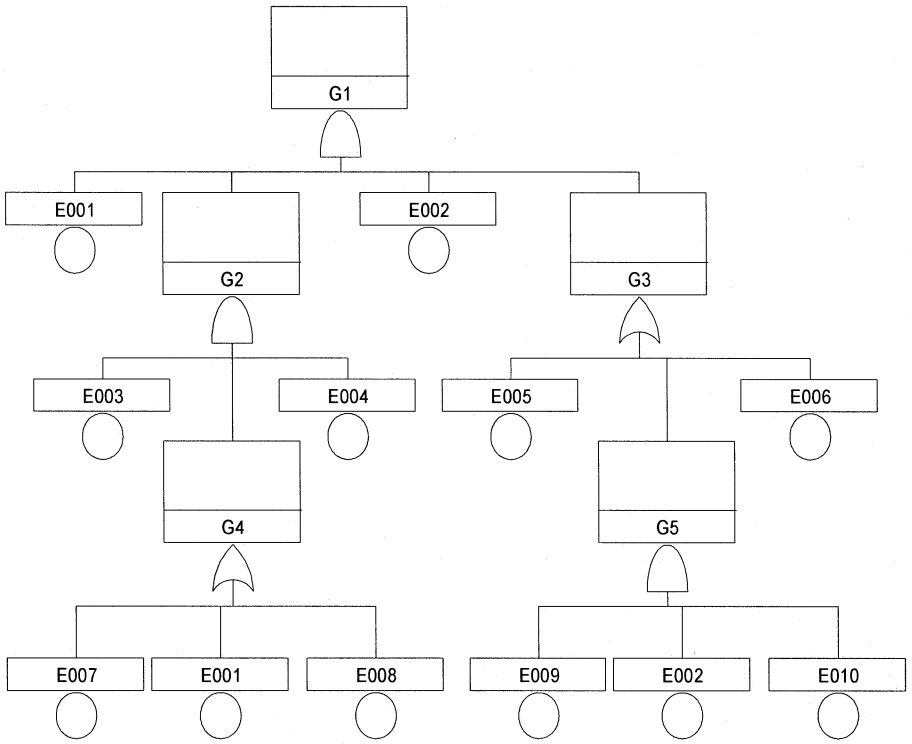

Figure 3.6: A fault tree with repeated events

In Fig. 3.6, the basic events E001, E002, E003, E004, E005, E006, E007, E008, E009 and E010 have failure probabilities .001, .004, .003, .007, .002, .006, .004, .003, .005 and.002 .In this example we have 2 repeated events E001 and E002 .E001 is an input to an OR gate G4 as well as AND gate G1 whereas E002 is an input to both AND gates G1 and G5. The top gate is an AND gate.

Steps 1 and 2 of the algorithm are shown in the tabular form (Table 11).

Gate	Step1	Step2
G4	t(G4)=OR, t(E007)=BE, t(E008)=BE, t(E001)=BE List(G4)(E007,.004,BE,NULL) (E008,.003,BE,NULL) (E001,.001,BE,NULL)	t(G4)=OR t(G2)=AND v(G4)=.008
G5	t(G5)=AND, t(E002)=BE, t(E009)=BE, t(E010)=BE List(G5)(E010,.002,BE,NULL) (E002,.004,BE,NULL) (E009,.005,BE,NULL)	t(G5)=AND t(G3)=0R v(G5)=.000000040
G2	t(G2)=AND, t(E003)=BE, t(E004)=BE, t(G4)=OR List(G2)(E003,.003,BE,NULL) (E004,.007,BE,NULL) (G4,.008,0R,(E007,.004,BE,NULL) (E008,.003,BE,NULL) (E001,.001,BE,NULL))	t(G2)=AND t(G1)=AND v(G2)=.003
G3	t(G3)=OR, t(E006)=BE, t(G5)=AND List(G3)(E006,.006,BE,NULL) (E005,.002,BE,NULL) (G5,AND,.000000040,(E010,.002,BE,NULL) (E002,.004,BE,NULL) (E009,.005,BE,NULL))	t(G3)=OR t(G1)=AND v(G3)=.008
G1	t(G1)=0R, t(E001)=BE, t(E002)=BE, t(G2)=AND, t(G3)=OR List(G1)**(E001,.001,BE,NULL)** (E003,.003,BE,NULL) (E002,.004,BE,NULL) (E004,.007,BE,NULL) (G4,008,OR,(E007,.004,BE,NULL) (E008,.003,BE,NULL) (E001,.001,BE,NULL)) (G3,.008,OR,(E006,.006,BE,NULL) (E005,.002,BE,NULL) (G5,AND,.000000040,(E010,.002,BE,NULL) (E002,.004,BE,NULL) (E009,.005,BE,NULL))	

Table 11: Algorithm steps for a fault tree with repeated events in Figure 3.6

Step 2 is not required at the top gate; therefore it is not shown in the table.

Step 3. t(G1) =AND, event E001 is an input to an AND gate G1 and OR gate G4 . We will traverse the List(G1), entries of the OR gate that contains the repeated basic event will be considered to have zero importance. From List(G1) we can see InputList of G4 contains the repeated event E001 and any other event of this gate is not repeated in another AND gate, therefore E007 and E008 will be have Zero importance .
Event E002 is repeated in both AND gates G1 and G5, therefore the importance of E009 and E010 is not affected.

Step 4. Most sensitive event is the top most event of List(G1) i.e. **E001.**

Table 12 contains the BIM measure computed using *Aralia.* According to BIM also, the most important component is **E001,** as it has the highest BIM.

Basic Event	Failure Probability	BIM
E001	**.001**	**6.72E-10**
E003	.003	2.24E-10
E002	.004	1.68E-10
E004	.007	9.60E-11
E006	.006	8.38E-11
E005	.002	8.35E-11
E010	.002	4.17E-13
E009	.005	1.67E-13
E007	.004	0.00E+00
E008	.003	0.00E+00

Table 12: BIM for a fault tree with repeated events in Figure 3.6

3.7 Summary

In this chapter we have discussed the algorithm to find the most sensitive component of the fault trees without as well as with repeated events. Some special cases of this algorithm were also considered. The results of these cases were compared with the Birnbaum importance measure. In the end we considered two examples one each for a fault tree with repeated and without repeated events. These examples showed that our algorithm finds the same most important component as the BIM computed using Aralia.

Chapter 4

Ranking

The algorithm for finding the most sensitive component described in chapter 3 has been modified in this chapter to rank all the components of the fault tree.

4.1 Ranking Algorithm

The first 3 steps of the algorithm stay the same as algorithm discussed in chapter 3. The modification has been done in step 4.

For ranking purpose we need to define a concept of *layers*. In List (G1), AND gate indicates the beginning of next layer. Within this layer when further AND gate is hit, a new layer starts.

First the events of the lowest layers are ranked followed by the higher layers. Once the events of a lowest layer are ranked, the next higher layer gates are arranged in descending (t(G1)) =OR or ascending (t(G1) =AND). The process is continued till the highest layer of gates (see example in 4.3.1, 4.3.2 and 4.3.3)

When two gates (same type) in the same layer have almost same gate value, combine the input lists of these to gate into one and arrange the events in ascending (AND gate) or descending (OR gate) order.

4.2 Special Cases

4.2.1: A fault tree with all AND gates

Basic events of the fault tree will be ranked according to the increasing order of their failure probabilities.

Let us consider the failure probabilities of basic events E001, E002, E003, E004, E005 and E006 in Figure 4.1 as .003, .004, .002, 005, .001 and .003 respectively. According to the above mentioned rule the basic events are arranged in the increasing order of the failure probabilities.

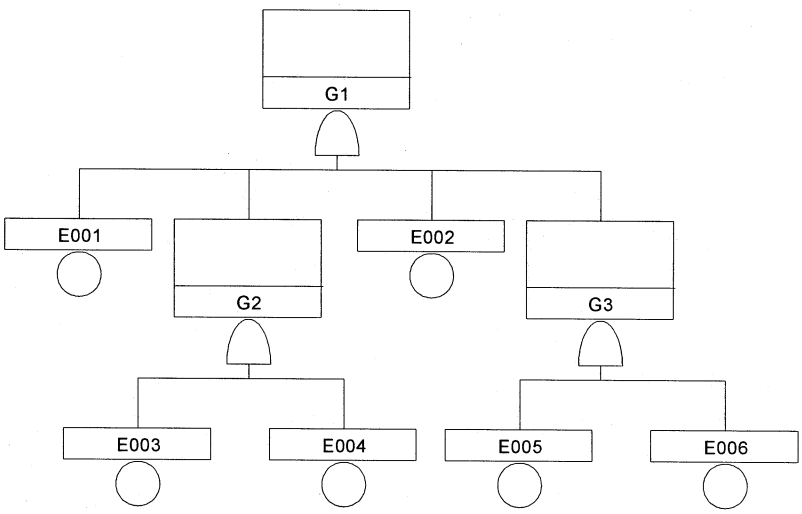

Figure 4.1: A fault tree with all AND Gates

The ordering according to the algorithm is shown in the 3rd column of the Table 13 and the ranking by Birnbaum importance measure is shown in the 5th column. From the table, we can see that the ranking done by our algorithm is same as that obtained by ordering the basic events based on their BIM.

Basic Event	Failure probability	Algorithm Ranking	BIM	BIM Ranking
E005	0.001	**1**	3.60E-13	**1**
E003	0.002	**2**	1.80E-13	**2**
E001	0.003	**3**	1.20E-13	**3**
E006	0.003	**3**	1.20E-13	**3**
E002	0.004	**4**	9.00E-14	**4**
E004	0.005	**5**	7.20E-14	**5**

Table13: Ranking of a fault tree with all AND gates in Figure 4.1

2.2: A fault tree with all OR gates

Basic events of the fault tree will be ranked according to the decreasing order of their failure probabilities.

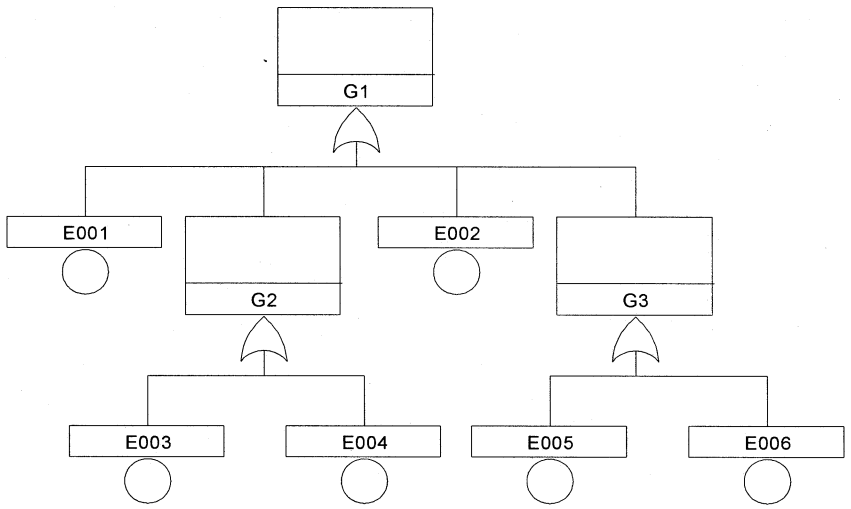

Figure 4.2: A fault tree with all OR gates

Considering the same failure probabilities as in section 4.2.1, the basic events will now be arranged according to the decreasing order of their failure probabilities. The ranking by our algorithm is shown in 3rd column of Table 14 and ranking by the Birnbaum Importance measure is in the 5th column. 4th column shows the BIM calculated by Aralia. We find that the ranking of basic events done by our algorithm matches with the one obtained by ordering the events based on their BIM measure.

Basic Event	Failure Probability	Algorithm Ranking	BIM	BIM Ranking
E004	0.005	**1**	9.87E-01	**1**
E002	0.004	**2**	9.86E-01	**2**
E001	0.003	**3**	9.85E-01	**3**
E006	0.003	**3**	9.85E-01	**3**
E003	0.002	**4**	9.84E-01	**4**
E005	0.001	**5**	9.83E-01	**5**

Table14: Ranking of a fault tree with all OR gates in Figure 4.2

4.3 Examples

In this section we have considered 3 examples. First we will consider a simpler example with top gate as AND gate to give reader the idea of layers. Then we will elaborate the ranking algorithm for a complex fault tree. Finally we will consider a case where our algorithm shows slight variation from Birnbaum importance measure.

4.3.1 A fault tree with top AND gate

The failure probabilities of events E001, E002, E003, E004, E005, E006 and E007 of the fault tree in Figure 4.3 are .007, .003, .005, .002, .001, .004 and .006 respectively.

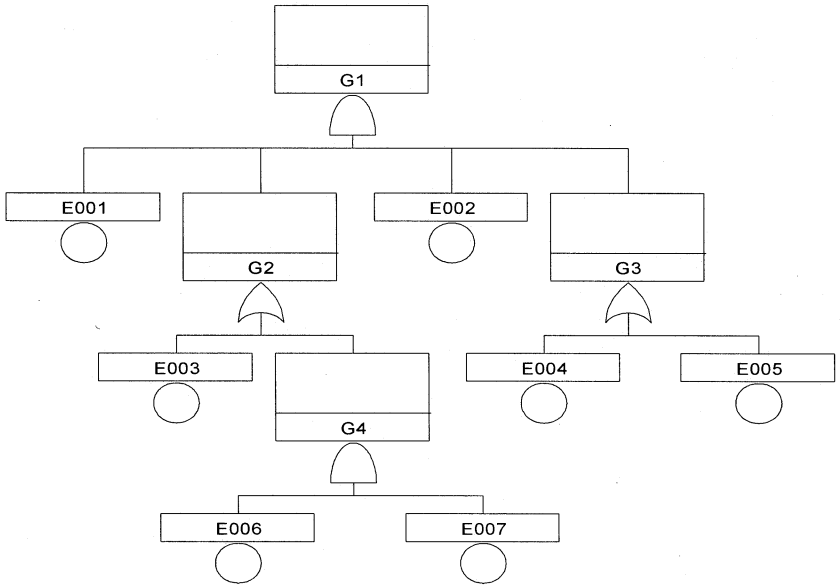

Figure 4.3: A fault tree with top AND gate

The steps of the Ranking Algorithm for fault tree in figure 4.3 are shown in table 15.

Gate	Step 1 –Create List	Step 2 –Gate Value
G4	t(G4)=AND, t(E006)=BE, t(E007)=BE List(G4)(E006,.004,BE,NULL) (E007,.006,BE,NULL)	t(G4)=AND t(G2)=OR v(G4)=.000024
G2	t(G2)=OR, t(E003)=BE , t(G4)=AND List(G2)(E003,.005,BE,NULL) (G4,.000024,AND,(E006,.004,BE,NULL) (E007,.006,BE,NULL)	t(G2)=OR, t(G1)=AND v(G2)=.0050024
G3	t(G3)=OR, t(E004)=BE, t(E005)=BE List(G3)(E004,.002,BE,NULL) (E005,.001,BE,NULL)	t(G3)=OR, t(G1)=AND v(G3)=.003
G1	Step 1 & 4 t(G1)=AND, t(E001)=BE, t(E002)=BE, t(G3)==OR, t(G2)=OR List(G1)(E002,.003,BE,NULL) **1** (G3,.003,OR,(E004,.002,BE,NULL) **2**	

	(E005,.001,BE,NULL))	**3**
	(G2,.005,OR,(E003,.005,BE,NULL)	**4**
	(G4,.000024,AND,(E006,.004,BE,NULL)	**6**
	(E007,.006,BE,NULL))	**7**
	(E001,.007,.BE,NULL)	**5**

Table 15: Steps of Ranking Algorithm for a fault tree with top AND gate in Figure 4.3

We now explain step 4 in the above Table 15. The List(G1) is shown in the last row of this table. E002, G3, G2, E001 are in layer 1 and G4 is in layer 2.Therefore first the basic events in layer 1 (E002, E004, E005, E003, E007) are ranked in the order they are arranged followed by the basic events in layer 2 (E006, E007). We have shown ranking along with the basic events in the List(G1).

Table 16 shows that ranking obtained from our algorithm matches exactly with the ordering of basic events based upon their BIM computed using Aralia.

Basic Event	Failure probability	Algorithm Ranking	BIM	BIM Ranking
E002	.003	**1**	1.05E-07	**1**
E004	.002	**2**	1.05E-07	**2**
E005	.001	**3**	1.05E-07	**3**
E003	.005	**4**	6.30E-08	**4**
E001	.007	**5**	4.52E-08	**5**
E006	.004	**6**	3.76E-10	**6**
E007	.006	**7**	2.51E-10	**7**

Table 16: BIM measure for a fault tree in Figure 4.3

4.3.2. A complex fault tree

Now we consider a complex example and show how this algorithm works efficiently for complex trees also. The basic events of the fault tree in Figure 4.4 are E001, E002, E003, E004, E005, E006, E007, E008, E009, E010, E011, E012, E013, E014, E015 and E016 with the failure probabilities .003, .001, .002, .004, .001, .002, .003, .005, .001, .002, .004,.003, .005, .005, .004 and .006 respectively.

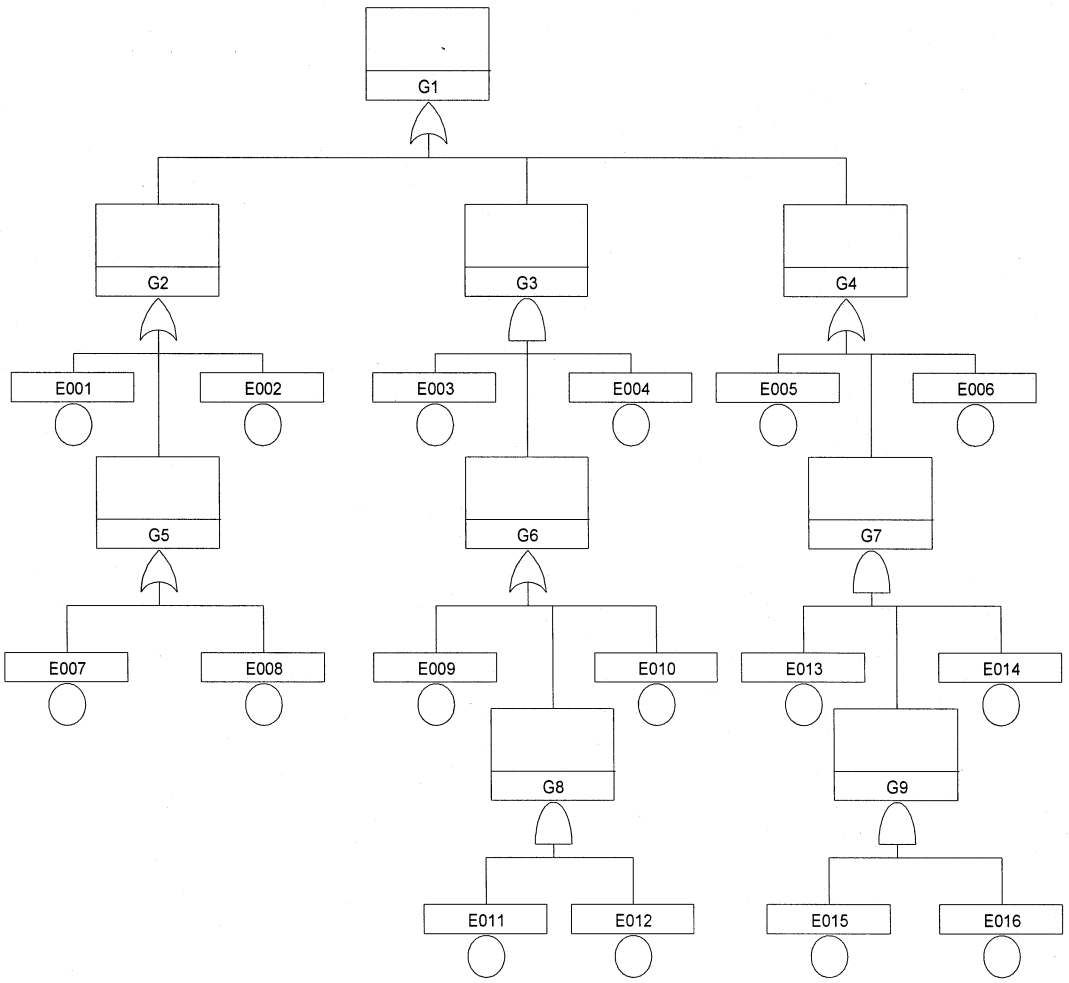

Figure 4.4: A complex fault tree

The following table shows the steps of ranking algorithm for a fault tree in Figure 4.4. The inputlist and value of the events is computed moving from the lowest level to the topmost level i.e. G1.Step 4 at the top gate is explained after this table.

Gate	Step 1	Step2
G8	t(G8) =AND, t(E011)=BE, t(E012)=BE List(G8)(E012,.003,BE,NULL) (E011,.004,BE,NULL)	t(G8)=AND t(G6)=OR v(G8)=.000012
G9	t(G9) =AND, t(E015)=BE, t(E016)=BE List(G9)(E015,.004,BE,NULL) (E016,.006,BE,NULL)	t(G9)=AND t(G7)=AND v(G9)=.004

G5	t(G5) =OR, t(E007)=BE, t(E008)=BE List(G5)(E008,.005,BE,NULL) (E007,.003,BE,NULL)	t(G5)=0R t(G2)=0R v(G5)=.005
G6	t(G6)=OR, t(E009)=BE, t(E010)=BE , t(G8)=AND List(G6)(E010,.002,BE,NULL) (E009,.001,BE,NULL) (G8,.000012,AND,(E012,.003,BE,NULL) (E011,.004,BE,NULL))	t(G6)=OR t(G3)=AND v(G6)=.003
G7	t(G7) =AND, t(E013)=BE, t(E014)=BE, t(G9)=AND List(G7)(E013,.002,BE,NULL) (E015,.004,BE,NULL) (E014,.005,BE,NULL) (E016,.006,BE,NULL)	(G7)=AND t(G4)=OR v(G7)= .00000000024
G2	t(G2)=OR, t(E001)=BE, t(E002)=BE, t(G5)=OR List(G2)(E008,.005,BE,NULL) (E001,.003,BE,NULL) (E007,.003,BE,NULL) (E002,.001,BE,NULL)	t(G2)=0R t(G1)=OR v(G5)=.005
G3	t(G3)=AND, t(E003)=BE, t(E004)=BE, t(G6)=OR List(G3) (E003,.002,BE,NULL) (G6,.003, 0R,(E010,.002,BE,NULL), (E009,.001,BE,NULL) (G8,.000012,AND, (E012,.003,BE,NULL), (E011,.004,BE,NULL)) (E004,.004,BE,NULL)	t(G3)=AND t(G1)=OR v(G3)= .000000024
G4	t(G4) =0R, t(E005)=BE, t(E006)=BE, t(G7)=AND List(G4)(E006,.002,BE,NULL) (E005,.001,BE,NULL) (G7,.000000000024,AND,(E013,.002,BE,NULL) (E015,.004,BE,NULL) (E014,.005,BE,NULL) (E016,.006,BE,NULL))	t(G4)=0R t(G1)=OR v(G4)=.002
G1	Step 1 , 3 and 4 t(G2)=0R, t(G3)=AND, t(G4)=OR List(G1) (E008,.005,BE,NULL) (E001,.003,BE,NULL) (E007,.003,BE,NULL)	 **1** **2** **2**

	(E006,.002,BE,NULL)	**3**
	(E005,.001,BE,NULL)	**4**
	(E002,.001,BE,NULL)	**4**
	(G3,.000024,AND, (E003,.002,BE,NULL)	**5**
	(G6,.003,0R,(E010,.002,BE,NULL)	**6**
	(E009,.001,BE,NULL)	**7**
	(G8,.000012,AND,(E012,.003,BE,NULL)	**13**
	(E011,.004,BE,NULL))	**14**
	(E004,.004,BE,NULL)	**8**
	(G7,.00000000024,AND,(E013,.002,BE,NULL)	**9**
	(E015,.004,BE,NULL)	**10**
	(E014,.005,BE,NULL)	**11**
	(E016,.006,BE,NULL))	**12**

Table 17: Steps of Ranking Algorithm for a fault tree in Figure 4.4

Now we explain step 4 for the fault tree in Figure 4.4 .The basic events E001, E002, E005, E006, E007, E008 are in layer 1, gate G3 and G7 form layer 2 and gate G8 is layer 3. Therefore events E003, E004, E009, E010, E013, E014, E015, E016 are in layer 2 are ranked after the first layer basic events. The basic events in G8 i.e. E011 and E012 are in layer 3 are ranked in the end.

Table 18 shows that the ranking of the basic events of the fault tree in Figure 4.3 exactly matches with the ordering of basic events according to the BIM calculated by Aralia.

Basic Event	Failure Probability	Algorithm ranking	BIM	BIM Ranking
E008	0.005	**1**	9.90E-01	**1**
E001	0.003	**2**	9.88E-01	**2**
E007	0.003	**2**	9.88E-01	**2**
E006	0.002	**3**	9.87E-01	**3**
E002	0.001	**4**	9.86E-01	**4**
E005	0.001	**4**	9.86E-01	**4**
E003	0.002	**5**	1.19E-05	**5**
E010	0.002	**6**	7.87E-06	**6**
E009	0.001	**7**	7.86E-06	**7**
E004	0.004	**8**	5.93E-06	**8**
E013	0.002	**9**	1.18E-07	**9**
E015	0.004	**10**	5.91E-08	**10**
E014	0.005	**11**	4.73E-08	**11**
E016	0.006	**12**	3.94E-08	**12**

E012	0.003	**13**	3.14E-08	**13**
E011	0.004	**14**	2.36E-08	**14**

Table 18: BIM for the fault tree in Figure 4.4

4.3.3 An Exceptional Case

Now we will discuss the case were our algorithm shows slight variation from the ranking of Birnbaum importance measure. It is the case when two gates (same type) in same layer have approximately same gate values. In such cases our algorithm may show slight difference in ranking. The failure probabilities of basic events E001, E002, E003, E004, E005, E006, E007, E008, E009, E010, E011, E012, E013, E014 and E015 are .003, .001, .002, .004, .001, .002, .003, .005, .001, .002, .004, .003, .005, .002, .004 and .006 respectively.

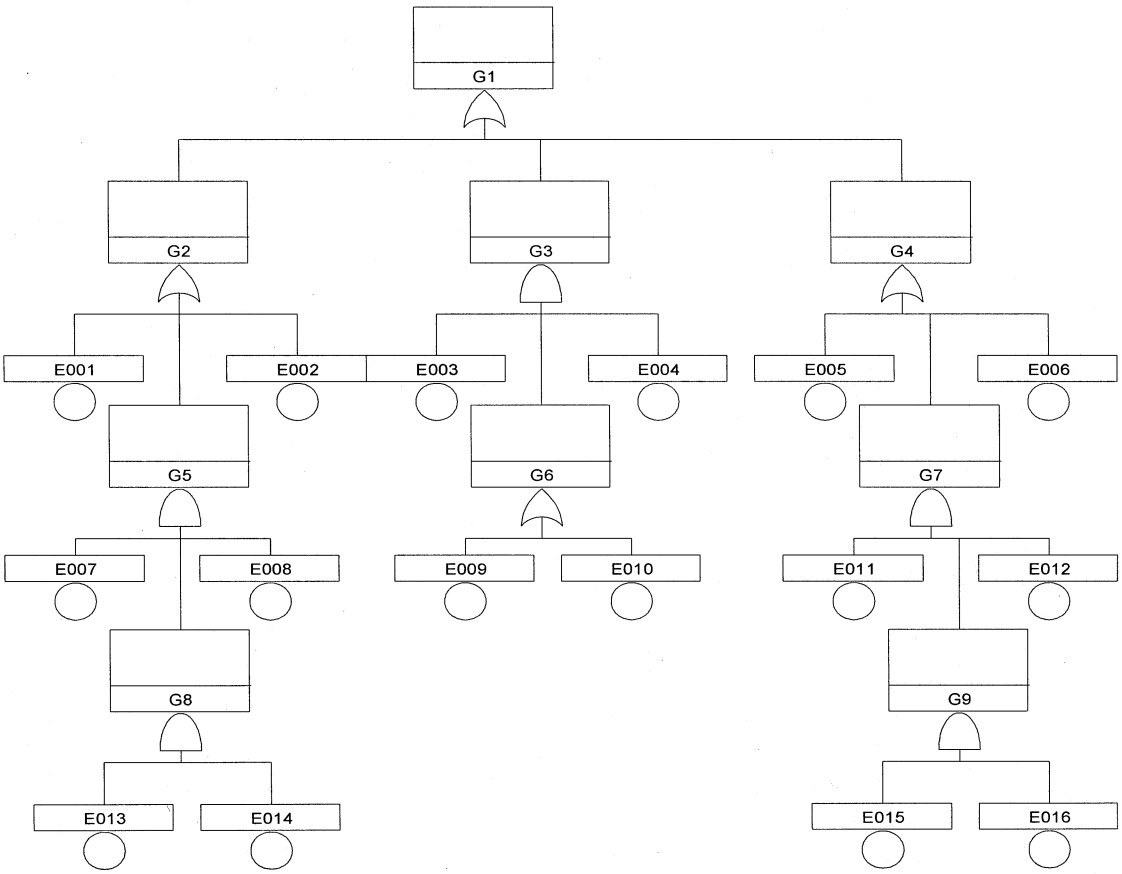

Figure 4.5: A fault tree with the lower AND gates having almost same gate values

Following table shows the steps of the algorithm for fault tree in Figure 4.5.

Gate	Step 1-create List	Step2 –Gate value
G8	t(G8) =AND, t(E013)=BE, t(E014)=BE List(G8)(E014,.002,BE,NULL) (E013,.005,BE,NULL)	t(G8)=AND t(G5)=AND v(G5)=.002
G9	t(G9) =AND, t(E015)=BE, t(E016)=BE List(G9)(E015,.004,BE,NULL) (E016,.006,BE,NULL)	t(G9)=AND t(G7)=AND v(G5)=.004
G5	t(E007)=BE,t(E008)=BE, t(G8)=AND ,t(G5)=AND List(G5)(E014,.002,BE,NULL) (E008,.003,BE,NULL) (E007,.005,BE,NULL) (E013,.005,BE,NULL))	t(G5)=AND t(G2)=OR v(G5)= .000000000150
G6	t(G6)=OR,t(E009)=BE,t(E010)=BE ,t(G8)=AND List(G6)(E010,.002,BE,NULL) (E009,.001,BE,NULL)	t(G6)=OR t(G3)=AND v(G6) =.003
G7	t(G7) =AND, t(E011)=BE,t(E012)=BE and t(G9)=AND List(G7)(E012,.003,BE,NULL) (E011,.004,BE,NULL) (E015,.004,BE,NULL) (E016,.006,BE,NULL)	(G7)=AND t(G4)=OR v(G7)= .000000000288
G2	t(G2) =OR, t(E001)=BE,t(E002)=BE,t(G5)=AND List(G2)(E001,.003,BE,NULL) (E002,.001,BE,NULL) (G5,.000000000150,AND,(E014,.002,BE,NULL) (E008,.003,BE,NULL) (E007,.005,BE,NULL) (E013,.005,BE,NULL))	t(G2)=OR t(G1)=OR v(G2)=.003
G3	t(G3)=AND, t(E003)=BE, t(E004)=BE ,t(G6)=OR List(G3)(E003,.002,BE,NULL) (G6,.003, 0R,(E010,.002,BE,NULL), (E009,.001,BE,NULL) (E004,.004,BE,NULL)	t(G3)=AND t(G1)=OR v(G3)= .000000024
G4	t(G4) =0R, t(E005)=BE,t(E006)=BE and t(G7)=AND List(G4)(E006,.003,BE,NULL)	t(G4)=0R t(G1)=OR

	(E005,.001,BE,NULL) (G7,.000000024,AND,(E012,.003,BE,NULL) (E011,.004,BE,NULL) (E015,.004,BE,NULL) (E016,.006,BE,NULL))	v(G4)=.003
G1	Step 1, 3 and 4 t(G2)=0R , t(G3)=AND, t(G4)=OR List(G1) (E001,.003,BE,NULL) **1** (E006,.003,BE,NULL) **1** (E002,.001,BE,NULL) **2** (E005,.001,BE,NULL) **2** (G3,.00000024,AND,(E003,.002,BE,NULL) **3** (G6,.003,0R,(E010,.002,BE,NULL), **4** (E009,.001,BE,NULL) **5** (E004,.004,BE,NULL) **6** (G7,.000000000288,AND,(E012,.003,BE,NULL) **7** (E011,.004,BE,NULL) **10** (E015,.004,BE,NULL) **10** (E016,.006,BE,NULL)) **12** (G5,.000000000150 ,AND,(E014,.002,BE,NULL) **8** (E008,.003,BE,NULL) **9** (E007,.005,BE,NULL) **11** (E013,.005,BE,NULL)) **11**	

Table 19: Steps of Ranking Algorithm for a fault tree in Figure 4.5

Last row of table 19 contains List(G1).The layer 1 basic events E001, E002, E005, E006 are ranked first .Gates G3, G7, and G5 form the second layer. They are ranked in the order they are arranged. But since G7 and G8 have approximately same value, the inputs list of both can be combined and the basic events can be ranked in the increasing order of their failure probabilities.

Basic Event	Failure Probabilities	Algorithm Ranking	BIM	BIM Ranking
E001	.003	1	9.95E-01	1
E006	.002	1	9.95E-01	1
E002	.001	2	9.93E-01	2
E005	.001	2	9.93E-01	2
E003	.002	3	1.19E-05	3
E010	.002	4	7.93E-06	4

E009	.001	5	7.92E-06	5
E004	.004	6	5.95E-06	6
E012	.003	7	9.52E-08	7
E014	.002	8	7.44E-08	8
E011	.004	**10**	7.14E-08	**9**
E015	.004	**10**	7.14E-08	**9**
E008	.003	**9**	4.96E-08	**10**
E016	.006	**12**	4.76E-08	**11**
E007	.005	**11**	2.98E-08	**12**
E013	.005	**11**	2.98E-08	**12**

Table 20: BIM measure for the fault tree in Figure 4.5

Table 20 shows that there is minor difference in the ranking of the basic events (E007, E008, E011, E013, E015 and E016) by our algorithm and the ordering done according the BIM. But these basic events are of the least significance, therefore the slight difference in their ranking do not affect an important decision.

4.4 Summary

In this chapter we have discussed the algorithm to rank the components of the fault tree. We considered special cases of the algorithm where we can rank the components just by looking at the tree structure. We then illustrated this algorithm with the help of simple and complex fault trees. In the end we discussed an exceptional case where our algorithm differs slightly from the ordering of the basic events based upon the Birnbaum importance measure calculated using Aralia.

Chapter 5

Conclusions

5. 1 Summary and future work

In order to improve the reliability of the system we first concentrated on finding the most sensitive component whose improvement will lead to the maximum improvement in the system reliability. For this purpose we developed an algorithm that finds such a component easily and accurately. The engineer may want to focus on more than one component that may also be playing a significant role in the system unreliability. Thus we extended our work by generalizing this algorithm for ranking all the components of the fault tree. The components are ranked according to their significance in the system unreliability.

When compared with computation of importance measures which involve finding partial derivatives, this algorithm involves simple arithmetic. It is much simpler and easy to use. To deal with repeated events without much effort is another asset of our work.

Overall, we have made a contribution to ease the improvement of system reliability by enabling the user to find the components which are affecting the reliability of the system most, without relying upon the software packages.

The algorithm for finding the most sensitive component works accurately for all the fault trees. The Ranking algorithm showed slight difference in few cases although it ranks the components of most of the fault trees accurately. In our future work we would like to refine our "Ranking" algorithm so that it works for all the tree structures.

5.2 Advantages of the proposed algorithms

Accuracy: The algorithm proposed in chapter 3 for finding the most sensitive component gives the same result as to those derived from the Birnbaum importance measure under the stated assumptions. The ranking algorithm proposed in chapter 4 is also accurate for most of the fault trees.

Ease of Application: The proposed methodology is quite simple and elegant in the sense that the user has only to follow a set of easy and clear guidelines to analyze and rank the basic events of the fault trees.

Computational Simplicity: The calculations whenever required, involve simple arithmetic. It cuts down a great deal of time. In certain cases such as the fault tree with all OR gates, all AND gates we can not only find the most sensitive component but accurately rank all the components of the tree just by looking at it.

Application to fault trees with repeated events: For a fault tree with repeated events, calculation effort increases significantly if the importance measures are to be calculated. But our algorithms handle the repeated events efficiently without an extra effort.

5.3 Limitations of proposed algorithms

The result of our ranking algorithm proposed in chapter 4 varies slightly from Birnbaum importance measure in certain tree structures when the gates of same type in the same layer have approximately same gate values (section 4.3.3).

Although we can argue for AND gates, as most of the fault trees consist of OR gates and may be few AND gates. AND gates imply that there are redundancies in the system. In most industries, redundancies are kept minimal unless necessary to avoid high maintenance and cost. But in future we would like to overcome this issue for both type of gates.

References

[1]Boudali H., Crouzen P., Stoelinga M., "*Dynamic Fault Tree Analysis Using Input/Output Interactive Markov Chains*" 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pp.708-717, 2007.

[2]Liu D., Zhang C., Xing W., Li R., Li H., "*Quantification of Cut Sequence Set for Fault Tree Analysis*" HPCC, pp 755-765, 2007.

[3]Remenyte R., Andrews J.D., "*A simple component connection approach for fault tree conversion to binary decision diagram*", Proceedings of the First International Conference on Availability, Reliability and Security 2006, pp. 449 - 457, 2006.

[4]Wang W., Loman J., Vassiliou P., "*Reliability importance of components in a complex system*", Proceedings of the Annual Reliability and Maintainability Symposium, 2004, pp 1-6, 2004.

[5]Jung-Hua Lo, Huang C.Y, Kuo S.Y, Lyu M. R. "*Sensitivity Analysis of Software Reliability for Component-Based Software Applications*", Proceedings of the 27th Annual International Conference on Computer Software and Applications, pp. 500, 2003.

[6]Beeson S., Andrews J.D., "*Birnbaum's measure of component importance for noncoherent systems*," IEEE Transactions on Reliability, vol. 52, issue 2, pp. 213 – 219 June.2003.

[7]Reay K.A., Andrews J.D, "*A Fault Tree Analysis Strategy Using Binary Decision Diagrams*", Reliability Engineering and System Safety, Volume 78, Number 1, pp. 45-56, October 2002.

[8]Gokhale, S.S., Trivedi, K.S, "*Reliability prediction and sensitivity analysis based on software architecture*" Proceedings 13th international Symposium on Software Reliability Engineering ISSRE 2002 pp. 64 – 75, 2002.

[9]Trivedi K.S, "*Probability and Statistics with Reliability, Queuing, and Computer Science Applications*", 2nd Edition, Prentice Hall, Englewood Cliffs, NJ USA 2001.

[10]Yong Ou, Dugan, J.B. "*Sensitivity analysis of modular dynamic fault trees*" Proceedings of the 4th Computer Performance and Dependability Symposium, pp. 35 – 43, 2000.

[11]Carot V.; Sanz J. "*Criticality and sensitivity analysis of the components of a system*" Reliability Engineering and System Safety, Volume 68, Number 2, pp. 147-152 May 2000.

[12]Muppala K, Fricks R.M., and Trivedi K.S, "*Techniques For System Dependability Evaluation*", *Computational Probability*, Kluwer Academic Publishers, The Netherlands, pp.445-480, 2000.

[13]Amari S.V, Dugan J.B, and Misra R.B, "*Optimal Reliability Design of Systems Subject to Imperfect Coverage*," IEEE Transactions on Reliability, pp. 275-284, Sept. 1999.

[14]Bolch G, Greiner S., Meer H.D, Trivedi K.S, "*Queueuing Networks and Markov Chains*", New York: John Wiley & Sons, 1998.

[15]Gulati R, Dugan J.B, "*A modular approach for analyzing static and dynamic fault trees*" Proceedings of the Annual Reliability and Maintainability Symposium1997, pp.57 – 63, 1997.

[16]Lee H.S., Lie C.H., Hong J.S., "*A computation method for evaluating importance-measures of gates in a fault tree*", IEEE Transactions on Reliability, Volume 46, Issue 3, pp, 360 – 365, Sep 1997.

[17]Meng F.C., "*Comparing the importance of system components by some structural characteristics*", IEEE Transactions on Reliability, Volume: 45, Issue: 1, pp.: 59-65, Mar 1996.

[18]Sahner R.A, Trivedi K.S., Puliafito A., "*Performance and reliability analysis of computer systems an example-based approach using the SHARPE*", Kluwer Academic Publishers, 1996.

[19]Rai S., Veeraghavan M., Trivedi K.S, "*A Survey of Efficient Reliability Computation using Disjoint Products Approach*", Networks, Vol. 25, pp. 147- 163, 1995.

[20]Boyd M.A.,Iverson D.L, "*Digraphs and fault trees: a tale of two combinatorial modeling methods*" .Proceedings of the Annual Reliability and Maintainability Symposium 1993, Issue, 26-28, pp. 220 – 226 ,Jan 1993.

[21]Sharma T.C., Bazovsky I., "*Reliability analysis of large system by Markov techniques*" Proceedings of the Annual Reliability and Maintainability Symposium, 1993. pp.260 – 267. Jan. 1993.

[22]Boyd M.A, "*Tutorial - What Markov Modeling Can Do For You: An Introduction*", Tutorial Notes for the Annual Reliability And Maintainability. Atlanta, GA, 1993.

[23]Andrews J.D, Moss T.R., "*Reliability and Risk Assessment*", Longman Scientific and technical, Essex, 1993.

[24]Henley E.J., Kumamoto H., "*Probabilistic Risk Assessment*", IEEE Press, 1992.

[25]Veeraghavan M., Trivedi K.S, "*An improved Algorithm for Symbolic Reliability Analysis*", IEEE Transactions on Reliability, R-40(3), pp. 347-358 August 1991.

[26]Lewis E.E, Boehm F., "*Monte Carlo simulation of complex system mission reliability*" Proceedings of the 21st conference on Winter simulation Washington, D.C., United States pp. 497 - 504 ,1989.

[27]Littlewood B., "*Forecasting Software Reliability*". Lecture Notes in Computer Science, No. 341. Berlin: Springer-Verlag , 1989.

[28]Blake J.T, Reibman A.L.,K. S. Trivedi ,"*Sensitivity analysis of reliability and performability measures for multiprocessor systems*" Proceedings of the 1988 ACM SIGMETRICS conference on Measurement and modeling of computer systems,1988.

[29]Sahner R.A, Trivedi K.S, "*Reliability Modeling using SHARPE*" IEEE transaction on reliability, R-36(2), pp.186-193, June 1987.

[30]Colburn C., "*The combinatorics of Network Reliability*" Oxford University Press, New York, NY, 1987.

[31]Xie M., "*On some importance measures of system components*", Stochastic Processes Applications 25, pp. 273-280, 1987.

[32]Natvig B., "*New light on measures of importance of system components*". *Scand. J. Star.* 12, pp.43-54, 1985.

[33]Goel A.L, Bastani F.B., "*Foreword: Software Reliability*", IEEE Transactions on Software Engineering, Volume 11, Issue 12, pp. 1409-1410, 1985.

[34]Abraham J.A, "*An Improved Algorithm for Network Reliability*", IEEE Transactions on Reliability, vol. R-28, pp. 58-61, April 1979.

[35]Butler D., "*An Importance Ranking for System Components Based upon Cuts*", Operations Research, Vol. 25, No. 5 (Sep. - Oct., 1977), pp. 874-879, 1977.

[36]Barlow R. E. and Proschan F., "*Importance of system components and failure tree events*". Stochastic Process Applications 3, pp.153-173, 1975.

[37]Fussell J., "*How to hand calculate system reliability characteristics*", Transactions on Reliability, vol. R-24, pp. 169–174, Aug. 1975.

[38]Birnbaum Z.W., "*On the Importance of Different Components in a Multicomponent System*", Multivariate Analysis – II, Edited by P. R. Krishnaiah, Academic Press, pp. 581-592, 1969.

[39]http://www.arboost.com/arlshop-page.htm

[40]http://www.relex.com/resources/overview.asp

[41]http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf