Research Article

Quality and Reliability Engineering International

(wileyonlinelibrary.com) DOI: 10.1002/qre.1245

Published online 19 September 2011 in Wiley Online Library

A New Ordering Method of Basic Events in Fault Tree Analysis

Hong-Zhong Huang*† Hua Zhang and Yanfeng Li

The ordering of basic events is critical to fault tree analysis on the basis of binary decision diagrams (BDDs). Many attempts have been made to seek an efficient ordering result with the aim of reducing the complexity of BDD. In this article, a new ordering method, namely, priority ordering method, is proposed. The new method takes into account not only the effects of the layers of fault tree but also the repeated events, the neighboring events, and the number of events under the same gate. According to these four effects, the priorities that sort the basic events of the fault tree are defined. The new method inherits the merits of structure-based and weight-based methods. It is able to evaluate the basic events on the basis of the structure-based method and the size of the subtree on the basis of the weighted-based method. Demonstrated by the examples, the proposed priority ordering method is superior to the existing ordering methods in terms of reducing the nodes in the BDD and improving the efficiency in transforming a fault tree to a BDD. Copyright © 2011 John Wiley & Sons, Ltd.

Keywords: fault tree analysis; variable ordering schemes; binary decision diagram; priority ordering method

1. Introduction

Fault tree analysis (FTA) is an important approach for system reliability analysis. It has been widely used in many applications and has played an important role for improving product safety and reliability over the last 20 years. However, the traditional FTA is based on the Boolean algebra and often causes the so-called *combinatorial explosion* issue when dealing with large and complex systems.

Many methods have been proposed to resolve this problem. One of the most successful approaches is the binary decision diagrams (BDDs)[1]. The BDD is an efficient way to deduce the Boolean algebra. It not only can quickly and effectively identify all the minimal cut sets in the fault tree but also can accurately calculate the occurrence probability of the top event, that is, the system failure probability [2–4]. Before converting the fault tree into a BDD, the ordering sequence of the basic events in the fault tree must be determined first. The sequence of the basic events can directly influence the size of a BDD. If the ordering schemes are different, the same fault tree can be converted into different BDDs. In general, the smaller the BDD is, the less computation time is required. Friedman and Supowit [5] has pointed out that the computational complexity of the system failure probability has a linear relationship with the number of nodes in the BDD. Therefore, it is important to develop an efficient ordering scheme that can lead to a more compact BDD structure with less number of nodes.

To improve the performance of existing ordering methods, we present in this article a new method, called *priority ordering method*, to rank the basic events. The new method considers the complex relationship between the basic events and orders the events on the basis of their own features. Examples indicate that the proposed priority ordering method performs better than the existing ordering heuristics in handling large BDDs and in converting fault trees into BDD.

The remainder of the article is organized as follows. Section 2 reviews existing ordering schemes and points out their shortcomings. Section 3 introduces the new proposed priority ordering method. Section 4 uses a numerical example to illustrate the application of the new method. Section 5 summarizes the advantages of the proposed method and future work.

2. The shortcomings of the existing ordering schemes

2.1. Existing ordering schemes

In recent years, considerable research efforts have been made to investigate the ordering methods. Some of the methods are based on evolutionary algorithms[5] and the others are based on fault tree search[8–10]. Since the evolutionary algorithm is quite complex

School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China

**Correspondence to: Hong-Zhong Huang, School of Mechatronics Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 611731, China.*

†*E-mail: hzhuang@uestc.edu.cn*

in programming and most of the parameters are set by experience, we chose the fault tree search algorithm because it follows simple rules and is easy to implement.

Fault tree search algorithms can be further classified into three types, that is, structural-based methods[6], weight-based methods[6–8], and integrated methods[9,10] that combine the structure-based and weight-based methods. Table I summarizes the existing ordering methods for each type.

A brief description of each ordering method listed in Table I is given as follows:

(1) The top–down, left–right method is produced by listing the basic events in a top–down, left–right manner from the original fault tree structure.
(2) The new top–down, left–right method is similar to the top–down, left–right method, which makes some modifications in dealing with repeated events. At each subtree, the basic event inputs are listed with repeated events in the front. If the gate has more than one repeated events, the event with the most repetitions is placed first. If the number of repetitions is the same, then they are placed from left to right.
(3) The depth-first method is implemented by first dividing the fault tree into several smaller subtrees and then giving each subtree a top–down, left–right search/ordering, proceeding from the first subtree of the top event.
(4) The new depth-first method differs from the depth-first method when dealing with repeated events. If the subtree has more than one repeated events, then the most repeated event is placed first. If the number of repetitions is identical, then they are placed from left to right.
(5) The priority depth-first method is the optimized version of the depth-first method. It is because the higher the layer of basic events, the greater the effect they have on the fault tree. Therefore, in the priority depth-first method, for subtrees that are located at the same layer, the subtrees containing only the basic events are considered first.
(6) The new priority depth-first method is the same as the priority depth-first method, except that the repeated events are considered first.
(7) The top–down-weighted method sets the weight of the top event to 1 and then equally distributes it to the subevents in the next layer. When we calculate the weights of the basic events, repeated events can have cumulative weights for their repetitions, and the sequence of the basic events is decided by their weights.
(8) The down–top-weighted method starts from the basic events, and each of the basic events has a weight of 1/2. Calculate the sum of the weight of the basic events from the bottom to the top of the fault tree following the "AND" gate and the "OR" gate and then order the basic events according to their weights.
(9) The structure importance method accumulates the importance of the basic events on the basis of the Birnbaums formula[7] and then sorts the basic events according to their importance.
(10) The progressive neighbor first ordering method is a progressive ordering method, which emphasizes the logical relationship of the basic event variables in a fault tree and gives the ordering priority to the neighbor events of an ordered variable.

All these methods have the imitations because they can obtain good ordering results for only certain types of fault trees. It is difficult to find a method that can guarantee good results regardless of the fault tree structures.

2.2. *Deficiency of the existing ordering schemes*

The aforementioned six structure-based ordering method order the basic events mainly on the basis of their locations in the fault tree, which may result in that the same fault tree can have different ordering schemes if drawn differently. Also, these methods do not consider the logical relationship between basic events.

The top–down-weighted method and down–top-weighted method order the basic events according to their weights, whereas the size of BDD is affected not only by the weight of the basic events but also by the relationship between them. Hence, these two methods may not necessarily lead to an optimal BDD structure.

The structure importance method orders the basic events after analyzing their structure importance, which is against the purpose of basic event ordering. Basic event ordering is to build a compact BDD to obtain the reliability index of the system, which includes the structure importance of basic events. Hence, this is self-contradictory.

The progressive neighbor first ordering method combines the characteristics of the structure-based and the weight-based methods. It only considers the effect of repeated events on the same layer events, which is not enough for the case when the relationship between events is complex. Also, this method typically involves intensive computation for complex fault trees.

Table I. The existing ordering schemes of basic events

Structure based	Weight based	Integrated type
Top–down, left–right	Top–down weighted	Progressive neighbor first ordering
New top–down, left–right	Down–top weighted	
Depth first	Structure importance	
New depth first		
Priority depth first		
New priority depth first		

To improve the performance of existing methods, we proposed a priority ordering method, which is able to take advantages of both structure-based and weight-based methods.

3. Priority ordering method

3.1. Some explanations and fault tree simplification

Before introducing the priority ordering method, we defined some concepts to facilitate presentation.

Two events are said to be brother events if they are at the same layer and under one gate. If one event is a gate event, then its brother event is called a brother gate event. As illustrated in Figure 1, *C* is the brother event of *B* but *D* is not, *D* is the brother event of *E*, and G_1 and G_2 are the brother gate events of *A*.

A child event is the event directly under a gate. In Figure 1, *B* and *C* are the child events of G_1 and *D* and *E* are the child events of G_2, whereas G_1, G_2, and *A* are the child events of *T*.

To reduce the computational complexity, the initial fault tree of many practical systems should be simplified by removing some unrelated events before converting the fault tree into a BDD. The simplification is mainly based on the exchange law and the absorption law of the Boolean algebra and shown in the following content.

In a subtree, on the basis of the exchange law of the Boolean algebra, if the type of the lower layer gate is the same as that of the higher layer gate, then delete the lower one and move its child events directly under the higher layer gate. It can be expressed by the following formula:

$$A + (B + C) = A + B + C$$
$$A \times (B \times C) = A \times B \times C$$

If a basic event's brother gate event contains the basic event itself, then delete the whole brother gate event, which is based on the absorption law of the Boolean algebra. It can be expressed as follows:

$$A \times (A + B) = A$$
$$A + A \times B = A$$

3.2. The principles of priority ordering method

In practical systems, the relationship between the basic events in a fault tree is quite complex. Hence, when ordering basic events, it is necessary to consider multiple factors to obtain the optimal result. Specifically, factors such as the number of layers, repeated events, and neighboring events and the number of basic events are considered and prioritized in our proposed method.

(1) Priority 1: number of layers. In the fault tree quantitative analysis, the Birnbaum importance (BI) is an important index to be considered. BI can be interpreted as the "defense in depth" of the fault tree with respect to a particular basic event, as presented by Lu and Jiang[11]. The greater the BI, the lower the defense in depth. In other words, the higher the layer where the basic event resides, the greater the effect they have on the top event. Therefore, when ordering basic event, the number of layers can be treated as the primary factor, and the subtree with less number of layers has a higher priority.

(2) Priority 2: repeated events. Basic events with repeated times can appear at many subtrees and may have an effect on these subtrees and their basic events. When calculating the influence of the basic events on the top event, repeated events can have a cumulative effect on it and may have a greater effect than that of the nonrepeated basic events on the same layer of a fault tree. However, repeated events are only considered from the basic event aspect and treated as more than one basic event. On the contrary, the number of layers is considered from subtree aspect. Therefore, the repeated event is less important than the number of layers and can be considered as the secondary factor when ordering the basic events. The event with larger number of repetition is assigned with a higher priority.

(3) Priority 3: neighboring events. When a basic event or gate event fails, its neighborhood will be influenced first. On the basis of this fact, a progressive neighbor first ordering method for the BDD conversion was proposed by Du and Sun[9]. This method can yield a good result than some traditional ordering methods. Nevertheless, the neighboring event of the ordered event only has

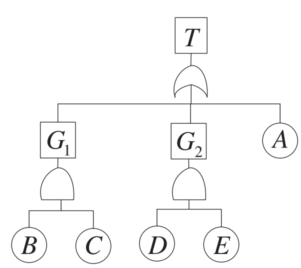

Figure 1. A fault tree used to illustrate the brother event and child event

a great influence on the basic events, and its influence on the top event may be much lower. Therefore, it is treated as the third factor in this article, and the neighboring event of an ordered event has a higher priority.

(4) Priority 4: number of basic events. The number of basic events does not mean one of the basic events of the whole subtree, but rather the basic events under the same gate at the same layer. If there are fewer basic events in the subtree, its influence on the top event may be greater than those with more basic events. This factor is also considered from the basic events point of view and may have less effect on the top event than the first three factors. As a result, it is considered as the fourth factor, and the subtree with fewer numbers of basic events has a higher priority.

(5) Priority 5: from left to right. If the previously mentioned factors are not applicable, the left–right scheme can be adopted to order the basic events.

The aforementioned five principles give the priority to the factors that influence the probability of the top events and their priority decreases from top to down.

Until now, we just set priorities to the factors and still do not know how to use it to order basic events in the FTA. To clearly describe the priority ordering method, we named and defined two rules, namely, the subtree selection rule and the basic event ordering rule. In practice, we first chose the subtree on the basis of the subtree selection rule and then ordered its basic events on the basis of the basic event ordering rule. After that, we chose another subtree and ordered its basic events, and so on.

(1) Subtree selection rule. First, on the basis of priority 1, we chose the subtree that is unordered and has the smallest number of layers. If two subtrees have the same number of layers, then on the basis of priority 2 choose the one for which the basic events have larger number of repetitions. If multiple events have the same number of repetitions, choose the subtree with the basic events that have been already ordered on the basis of priority 3. If the number of repetitions is still identical, select the subtree that has the least number of events according to priority 4. Further ties can be broken on the basis of priority 5 by choosing the left subtree to order.

(2) Basic event ordering rule. First, on the basis of priority 2, the basic events with the most repetitions are the event to be placed first. If the repetitions are same, then arrange them in accordance with priority 5 from left to right.

The process of the priority ordering method is shown in Figure 2, and l is used to take the record of the layers that the search process is in.

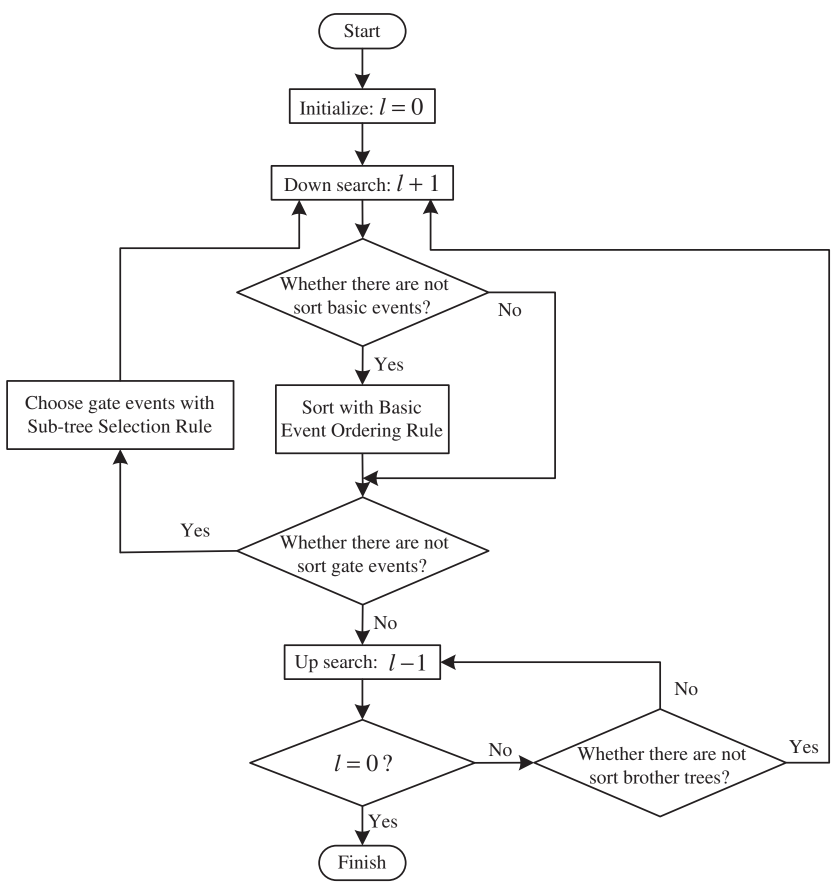

Figure 2. Flow chart of the priority method

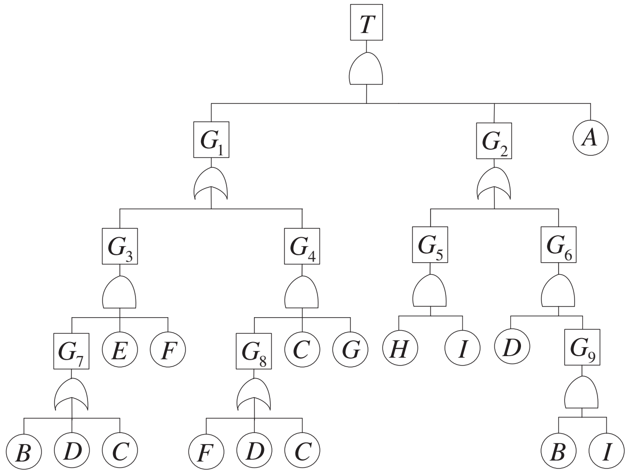

Figure 3. A fault tree used to illustrate priority ordering method

4. Case study

4.1. Example

To help better understand this method, the fault tree shown in Figure 3 is given as an example to illustrate the priority ordering method.

Before the application of the priority ordering method, the fault tree should be simplified because the child events of G_8 contain the basic event *C* and the brother events of G_8 also contain *C*. Hence, the whole gate event of G_8 is deleted on the basis of the absorption law of the Boolean algebra:

$$(F + D + C) \times C \times G = C \times G$$

G_6 is an "AND" gate, and its child event G_9 is also an "AND" gate. G_9 is hence deleted. Its subevents *B* and *I* are moved up and become the child events of G_6 on the basis of the exchange law of the Boolean algebra:

$$D + (B + I) = D + B + I$$

Figure 4 shows the simplified fault tree model.

The specific steps to order the basic events of the simplified fault tree using the priority ordering method are as follows:

(1) In this fault tree, the child event of the top event only has one basic event. We therefore chose the basic event and obtain the first order, φ_1={*A*}.
(2) G_1 and G_2 are two brother gate events of *A*. Comparing their sizes, G_2 only has two layers whereas G_1 has three. Therefore, G_2 is smaller and ordered first.

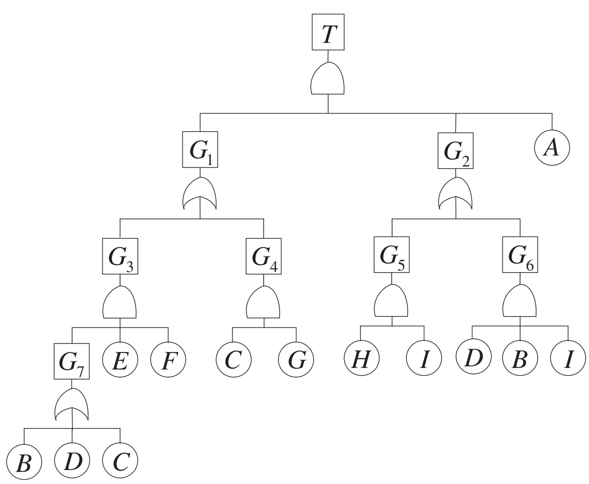

Figure 4. The simplified fault tree of the tree in Figure 3

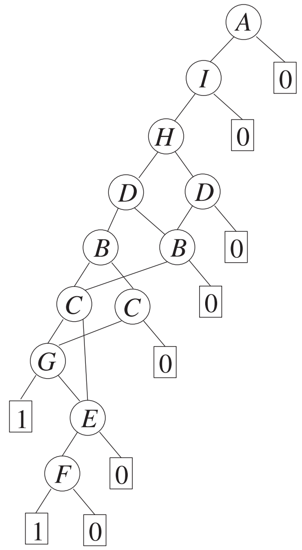

Figure 5. The binary decision diagram generated using the order obtained from priority ordering method

(3) The child events of G_2 contain two gate events, G_5 and G_6. By comparing, one can find that both of the two gate events only have one layer, which cannot be compared in accordance with priority 1. Then through priority 2, one can find that both of the two gate events contain the repeated event *I*; hence, this priority also does not work. Next, use priority 3, that is, the neighbor first ordering principle, to compare G_5 and G_6. Since neither has the basic event that has already been ordered, priority 3 is not applicable either. Then in accordance with priority 4, G_5 has two child events and G_6 has three; G_5 is ordered first accordingly. The child event *I* in G_5 is a repeated event and has a higher priority. The sequence $\varphi_2=\{I,H\}$ is hence obtained. Then order G_6; the repeated event *I* is ordered first. The remaining basic events *D* and *B* are ordered according to priority 5, from left to right. This results in the sequence $\varphi_3=\{I,D,B\}$.

(4) Next, choosing from the remaining brother gate events of G_2, we have G_1. Its child events has two gate events G_3 and G_4. G_3 has two layers and G_4 only has one. On the basis of priority 1, G_4 takes the priority, and this results in the sequence $\varphi_4=\{C,G\}$. Gate event G_3 has one gate event and two basic events. Choose the basic events and order them with priority 5, from left to right. This results in the sequence $\varphi_5=\{E,F\}$. Then order the gate event. Because all the child events of G_7 have been already ordered, the ordering is complete.

Finally, the sequence of basic events is shown as follows (in the formula, "$A<I$" means *A* is ordered before *I*).

$$A < I < H < D < B < C < G < E < F$$

Then through the Shannon decomposition[12], the BDD is generated using the previous order, as shown in Figure 5.

4.2. Comparison of the priority ordering method and other ordering methods

To show the advantage of the priority ordering method, we used the fault tree in Figure 4 as an example and ordered it with the existing 10 basic event ordering schemes listed in Table I. The ordering results are compared with that generated using our proposed priority ordering method.

Table II. Ordering sequence generated from various ordering methods

Methods	Ordering of basic events	No. BDD nodes
Top–down, left–right	$A<E<F<C<G<H<I<D<B$	16
New top–down, left–right	$A<I<E<F<C<G<H<D<B$	13
Depth first	$E<F<C<G<B<D<H<I<A$	16
New depth first	$E<F<C<G<B<D<I<H<A$	16
Priority depth first	$C<G<E<F<B<D<H<I<A$	16
New priority depth first	$C<G<E<F<B<D<I<H<A$	17
Top–down weighted	$A<I<C<G<H<D<B<E<F$	16
Down–top weighted	$C<G<B<D<E<F<I<H<A$	12
Progressive neighbor first ordering	$A<E<F<B<D<C<G<I<H$	19
Priority ordering	$A<E<F<B<D<C<G<I<H$	12

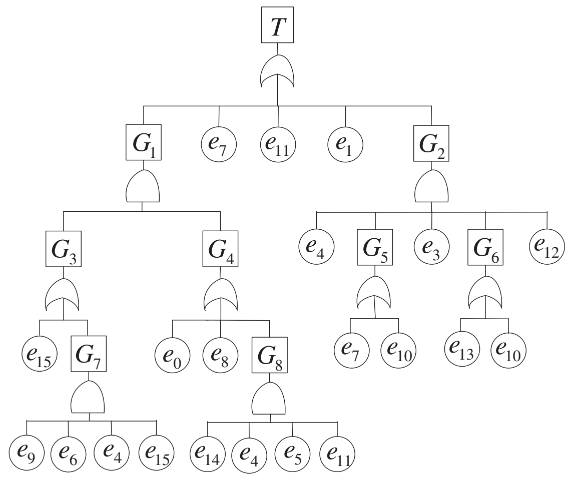

Figure 6. Fault tree as presented by Du and Sun [9]

Similarly, using the other methods in Table I to order the basic events of the fault tree, the results are shown in Table II. The nodes of the BDD mentioned in the next paragraph refer to the nonterminal nodes only.

From Table II, we can see that if the progressive neighbor first ordering method is applied, the number of nodes reaches 19. The number of BDD nodes using the priority ordering method is 12, which is the smallest among all the existing methods. Notice that the down–top-weighted method also leads to a 12-node BDD structure. To further demonstrate the merits of the priority ordering method, we provided further comparison in the next section.

4.3. *Further comparison*

In this section, another fault tree, which is originally from Du and Sun[9] and shown in Figure 6, is taken as an example.

The results of basic event ordering and nodes numbers are listed in Table III. From it, one can find that the priority ordering method yields the best ordering result in terms of the number of nodes in the BDD.

Other examples are also used for testing and the results show that the priority ordering method always yields the best ordering sequence.

Table III. Various ordering results of the fault tree [9]

Methods	Ordering of basic events	No. BDD nodes
Top–down, left–right	$e_7 < e_{11} < e_1 < e_4 < e_3 < e_{12} < e_{15} < e_0 < e_8 < e_{10} < e_{13} < e_9 < e_6 < e_{14} < e_5$	13
New top–down, left–right	$e_7 < e_{11} < e_1 < e_4 < e_3 < e_{12} < e_{15} < e_{10} < e_0 < e_8 < e_{13} < e_9 < e_6 < e_{14} < e_5$	12
Depth first	$e_7 < e_{11} < e_1 < e_{15} < e_9 < e_6 < e_4 < e_0 < e_8 < e_{14} < e_5 < e_3 < e_{12} < e_{10} < e_{13}$	13
New depth first	$e_7 < e_{11} < e_1 < e_{15} < e_4 < e_9 < e_6 < e_0 < e_8 < e_{14} < e_5 < e_3 < e_{12} < e_{10} < e_{13}$	13
Priority depth first	$e_0 < e_8 < e_{14} < e_4 < e_5 < e_{11} < e_{15} < e_7 < e_1 < e_3 < e_{12} < e_{10} < e_{13} < e_9 < e_6$	12
New priority depth first	$e_0 < e_8 < e_4 < e_{14} < e_5 < e_{11} < e_{15} < e_7 < e_1 < e_3 < e_{12} < e_{10} < e_{13} < e_9 < e_6$	12
Top–down weighted	$e_7 < e_{11} < e_1 < e_{15} < e_4 < e_{10} < e_3 < e_{12} < e_0 < e_8 < e_{13} < e_9 < e_6 < e_{14} < e_5$	14
Down–top weighted	$e_7 < e_{11} < e_1 < e_0 < e_8 < e_4 < e_{14} < e_5 < e_{15} < e_3 < e_{12} < e_{10} < e_{13} < e_9 < e_6$	12
Progressive neighbor first ordering	$e_7 < e_{11} < e_1 < e_{10} < e_{13} < e_4 < e_3 < e_{12} < e_{14} < e_5 < e_0 < e_8 < e_{15} < e_9 < e_6$	10
Priority ordering	$e_7 < e_{11} < e_1 < e_4 < e_3 < e_{12} < e_{10} < e_{13} < e_{15} < e_0 < e_8 < e_{14} < e_5 < e_9 < e_6$	10

5. Conclusions

The ordering scheme of the basic events has a significant effect on the FTA based on BDD. The various types of ordering schemes have been proposed in the literature to order the basic events, each having some advantages and disadvantages. The structure-based methods do not consider the logical relationship among the basic events, whereas the weight-based methods consider only the relationship between the weights of the basic events. The recently proposed progressive neighbor first ordering method only considers the effect of repeated events on the same layer events. None of these methods can grasp the complex relationship between interconnected events.

On the basis of the existing methods, a new method, namely, the priority ordering method, is presented to order the basic events. This method orders the basic events in accordance with the multilevel priorities, that is, the number of layers, repeated events, and neighboring events and the number of events in one gate. These four factors are taken into account during the ordering process to minimize the number of nodes in the final BDD structure and to improve the efficiency of the fault tree-to-BDD conversion. Besides, the proposed method combines the characteristics of the structure-based and weight-based methods. Specifically, the consideration of the repeated events and neighboring events is based on the structure-based methods, and the consideration of the number of layers and the number of events in one gate is based on the weight-based methods.

As illustrated by several examples, the proposed priority ordering method not only can generate a good ordering sequence, leading to an effective conversion from fault trees to BDD, but also has the general applicability, allowing good sorting results for fault trees. Nevertheless, it still has some deficiencies. For instance, the different drawings of the same fault tree may result in different ordering sequence. In this article, only four major factors that have an effect on the top event are considered. Whether other factors will affect the probability of the top event needs to be explored in the future work. Besides, the priority ordering method is only used in the binary state systems in this article; its application to complex systems, such as phased mission systems[13,14] and multistate systems[15–19], is the focal area of the future work.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under the contract number 50775026 and by the Specialized Research Fund for the Doctoral Program of Higher Education of China under the contract number 20090185110019.

References

1. Akers SB. Binary decision diagram. *IEEE Transaction on Computer* 1978; **27**(6):509–516.
2. Sinnamon RM, Andrews JD. Fault tree analysis and binary decision diagrams. IEEE Proceedings Annual Reliability and Maintainability Symposium 1996; 215–222.
3. Andrews JD. Event tree analysis using binary decision diagrams. Annual Reliability and Maintainability Symposium 2006; 1–12.
4. Sinnamon RM, Andrews JD. New approaches to evaluating fault trees. *Quality and Reliability Engineering International* 1997; **58**:89–96.
5. Friedman SJ, Supowit KJ. Finding the optimal variable ordering for binary decision diagrams. *IEEE Transactions on Computers* 1990; **39**(5):710–713.
6. Andrews JD, Bartlett LM. Efficient basic event orderings for binary decision diagrams. Proceedings Annual Reliability and Maintainability Symposium 1998; 61–68.
7. Bartlett LM, Andrews JD. Choosing a heuristic for the "fault tree to binary decision diagram" conversion using neural networks. *IEEE Transactions on Reliability* 2002; **51**(3):344–349.
8. Bartlett LM, Andrews JD. An ordering heuristic to develop the binary decision diagram based on structural importance. *Reliability Engineering and System Safety* 2001; **72**:31–38.
9. Du SG, Sun Y. A novel ordering method of binary decision diagram. International Conference on Management Science and Engineering 2007; 299–304.
10. Du SG, Sun Y. Comparison of progressive variable ordering methods with fixed ordering heuristics for binary decision diagrams. Wireless Communications, Networking and Mobile Computing 2007; 4581–4584.
11. Lu L, Jiang J. Joint failure importance for noncoherent fault trees. *IEEE Transactions on Reliability* 2007; **56**(3):435–443.
12. Coudert O, Madre J. Fault tree analysis: 10^{20} prime implicants and beyond. Proceedings of Annual Reliability and Maintainability Symposium 1993; 240–245.
13. Xing L, Dugan JB. Comments on PMS BDD generation. *IEEE Transactions on Reliability* 2004; **53**(2):169–173.
14. Xing L, Dugan JB. Analysis of generalized phased mission system reliability, performance and sensitivity. *IEEE Transactions on Reliability* 2002; **51**(2):199–211.
15. Huang J, Zuo MJ. Dominant multi-state systems. *IEEE Transactions on Reliability* 2004; **53**(3):362–368.
16. Ding Y, Zuo MJ, et al. The Hierarchical weighted multi-state k-out-of-n system model and its application for infrastructure management. *IEEE Transactions on Reliability* 2010; **59**(3):593–603.
17. Ding Y, Zuo MJ, *et al*. A framework for reliability approximation of multi-state weighted k-out-of-n systems. *IEEE Transactions on Reliability* 2010; **59**(2):297–308.
18. Liu Y, Huang HZ. Optimal selective maintenance strategy for multi-state systems under imperfect maintenance. *IEEE Transactions on Reliability* 2010; **59**(2):356–367.
19. Liu Y, Huang HZ. Optimal replacement policy for multi-state system under imperfect maintenance. *IEEE Transactions on Reliability* 2010, **59**(3):483–495.

Authors' biographies

Hong-Zhong Huang is a professor and the Dean of the School of Mechanical, Electronic, and Industrial Engineering, University of Electronic Science and Technology of China. He has held visiting appointments at several universities in the USA, Canada, and Asia. He received a PhD degree in reliability engineering from Shanghai Jiaotong University, China, and has published 150 journal papers and 5 books in fields of reliability engineering, optimization design, fuzzy sets theory, and product development. He is a Fellow of the International Society of Engineering Asset Management and a member of the European Safety and Reliability Association Technical Committee on System Reliability. He is a regional editor of the *International Journal of Reliability and Applications*. He is an editorial board member of the *International Journal of Reliability, Quality and Safety Engineering*, the *International Journal of Quality, Statistics, and Reliability*, the *International Journal of Reliability and Quality Performance*, the *Advances in Fuzzy Sets and Systems*, and the *Open Mechanical Engineering Journal*. He received the William A. J. Golomski Award from the Institute of Industrial Engineers in 2006, and the Best Paper Award of the 8th International Conference on Frontiers of Design and Manufacturing in 2008. His current research interests include system reliability analysis, warranty, maintenance planning and optimization, and computational intelligence in product design.

Hua Zhang is a master's degree student of the School of Mechanical, Electronic, and Industrial Engineering, University of Electronic Science and Technology of China. His research interests include system reliability analysis, and reliability-based design.

Yanfeng Li is a PhD candidate in the School of Mechanical, Electronic, and Industrial Engineering, University of Electronic Science and Technology of China. His research interests include system reliability analysis, reliability-based design, and life cycle reliability-based design.