Research Article

Reliability Analysis of Dynamic Fault Tree Based on Binary Decision Diagrams for Explosive Vehicle

Guang-Jun Jiang,1,2 Zong-Yuan Li,1 Guan Qiao,1,2 Hong-Xia Chen,1,2 Hai-Bin Li,1 and Hong-Hua Sun1,2

1School of Mechanical Engineering, Inner Mongolia University of Technology, Hohhot 010051, Inner Mongolia, China
2Inner Mongolia Key Laboratory of Advanced Manufacturing Technology, Hohhot 010051, Inner Mongolia, China

Correspondence should be addressed to Guan Qiao; qiaoguan@imut.edu.cn

Received 26 February 2021; Revised 24 March 2021; Accepted 29 March 2021; Published 10 April 2021

Academic Editor: Mohammad Yazdi

Copyright © 2021 Guang-Jun Jiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Dynamic fault tree is often used to analyze system reliability. The Markov model is a commonly used method, which can accurately reflect the relationship between the state transition process and the dynamic logic gate transfer in the dynamic fault tree. When the complexity or scale of system is increasing, the Markov model encountered a problem of state space explosion leading to increase troubles. To solve the above problems, a modular approach is needed. Based on the modular approach, a hybrid fault module was researched in this paper. Firstly, the stackable fault subtree containing complex static/dynamic logic gate is transformed into four common combinational logic gates through preprocessing of the dynamic gate in the module. When, the complexity of the model was reduced by incorporating four common combinational logic gates and using the binary decision graph to solve variable ordering in the calculation of failure probability of static subtree. Moreover, the calculating process of complex mixed logic gate fault tree can be simplified. An example of the ammonium nitrate/fuel explosive production system for BCZH-15 explosive vehicle was used to verify the feasibility of the presented method.

1. Introduction

Fault tree analysis (FTA) is a common method for the reliability modeling and evaluating large safety-critical systems as discussed in [1, 2]. The classic fault tree is a static tool, and its primary objective is to improve the safety for the probabilistic safety assessment as discussed in [3, 4]. Dynamic fault tree is an extension of common fault tree, which adds many special logic gates to express the existing problems in specific systems. Mi et al. [5] studied system action of mechatronics systems through taking the advantages of the dynamic fault tree (DFT). Fault tree analysis is a risk assessment method widely used in the process industry. Yazdi et al. [6–8] improved the traditional fault tree analysis, mainly for the study of uncertainty handling in qualitative and quantitative risk assessment procedures. The Markov model is appropriate for modeling complex systems that have traits of timing, sequencing, repair, redundancy, and fault tolerance as discussed by Ericson [9]. However, the Markov model becomes unwieldy as the system size increases. It is difficult to construct and solve Markov models for the large systems as discussed in [10–12]. Li [13] used the Bayesian network method to avoid dealing with Markov state explosion problem. Li et al. [14–16] used multiple methods such as multisource information fusion and Bayesian networks in the process of studying the reliability of offshore wind turbines. Dugan et al. [17–19] utilized dynamic gates to discern independent dynamic subtrees and used the different Markov model for each subtree through the process of modularization. As discussed by Jiang et al. [20], in solving the problem of reliability analysis and analysis of dynamic systems, fuzzy set theory is used. Huang et al. [21] proposed an improved decomposition scheme, where the dynamic subtree can be further modularized (if there exist some independent subtrees). A huge fault tree may contain dynamic subtrees and static subtrees. Use the
binary decision diagram (BDD) method to calculate the failure rate of the top event of the fault tree. BDD was applied to efficiently realize reversible functions as discussed by Mathias Soeken et al. [22]. The modularization for large fault trees before analysis can improve computational efficiency. The approach of dynamic innovative fault tree (DIF tree) as discussed in [23] exploited the modularization by combining static and dynamic analysis. Ge et al. [24] analyzed DFT using improved the sequential binary decision diagram (SBDD) method. Bandyopadhyay et al. [25] improved a design technique, where BDDs are used to construct optical circuits.

However, after the modularization, the applicability of the model should be considered for relatively complex systems. Zhang et al. [26] proposed a generalized dynamic reliability model to calculate reliability of system under complex load. In dynamic systems, the semi-Markov process (SMP) was applied to solve the problem that the lifetime of components follows nonexponential distributions as discussed by Li et al. [27]. Abimbola et al. [28] presented a risk assessment methodology for analyzing the safety of critical components based on BN as discussed in [28]. This paper studies how to use modular methods to solve system reliability problems.

In recent years, the focus on field mixed explosive vehicle research has been performed to improve or design new electrical control systems to enhance the safety of equipment; to study the advantages of equipment in special geographic locations; to improve the blasting efficiency by changing the composition of the mixed explosive. There are only a few literatures on the reliability of the equipment, and only the reliability of the equipment can meet the operating requirements and can the safety, superiority, and efficiency of the equipment be demonstrated. With the dynamic characteristics becoming more and more obvious, it is urgent to evaluate equipment reliability considering the dynamic characteristics.

The remainder of the paper is organized as follows: Section 2 introduces division method of fault tree modular and illustrates how to preprocess the dynamic fault tree’s logical gates. Section 3 investigates the static subtree BDD theory solution method. The dynamic fault tree Markov theory solution method is described in Section 4. The proposed method was applied to calculate the reliability of the explosive production system for the BCZH-15 explosive vehicle in Section 5. Finally, the conclusion of the article is presented.

### 2. Fault Tree Modular Analysis Method

#### 2.1. Modular Division for Fault Tree

Rauzy et al. [29] presented the fault tree depth traversal method, which can modularize the fault tree. This method is concise and highly effective to deal with fault tree modularization problem. The deep traversal node method is used to divide module for the fault tree, which is easy to discover modules and zone. The procedure is as follows:

1. Calculate the traverse node for the fault tree and list the results
2. Sort the root node, leaf nodes, and the intermediate nodes and perform depth-first leftmost traversal for them
3. Collect the information of the internal event \( V \) for the third traversal node and collect the first and the last period of each subtime to provide strong evidence for modularizing the event.

Module division for the fault tree is shown in Figure 1, and the results are shown in Tables 1 and 2.

In Table 1, \( S \) indicates that the fault tree executes the left traversal node rule and then passes through the number of the node in turn, and \( V \) represents the name of each node in the fault tree. The sequence is given in Table 1.

Table 1 can summarize the target node numbers that each module can reach after executing the leftmost traversal node method, and then push out whether the modules are independent. In Table 2, \( \text{Min} \) is the node number that the module first arrives, and \( \text{Max} \) is the last module. \( Y \) means that the module is a separate module, and \( \text{N} \) means a nonindependent module. The division results are shown in Table 2.

From the operations given in Table 2, it can be concluded that the part of the fault tree modulo can be divided into \( T, M_1, M_4, M_5, \) and \( M_6 \).

#### 2.2. Preprocessing of Logic Gate

After modularizing the fault tree, it is necessary to further determine the module state (dynamic or static). The BDD method is used to calculate the subtree for the static submodule. The Markov model is applied to mold for the dynamic submodule. Since the system is divided into small modules, state explosion problem will not occur during the model processing. Due to logic gate nesting, the static submodule may contain dynamic submodule after the module division in a complex fault tree. In this case, the corresponding algorithm cannot be applied to the fault tree submodule. Preprocessing of dynamic logic gate is required for the basic fault tree, and the complex hybrid dynamic logic gate is transformed into an easy solution model, which can greatly simplify the subsequent calculation.

##### 2.2.1. Processing of Simple Logic Gate

Pretreatment of the logic gate includes the following: copretreatment of AND gate by OR gate, pretreatment of AND gate or PAND gate, pretreatment of two kinds of PAND gate stacking, and pretreatment of FDEP gate (Figures 2–5).

The above is the basic preprocessing process of logic gates.
Table 1: First traversal result.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>T</td>
<td>M1</td>
<td>M2</td>
<td>M4</td>
<td>A1</td>
<td>A2</td>
<td>M4</td>
<td>M5</td>
</tr>
<tr>
<td>S</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>V</td>
<td>A3</td>
<td>A4</td>
<td>M5</td>
<td>M2</td>
<td>M3</td>
<td>M5</td>
<td>M6</td>
<td>A5</td>
</tr>
<tr>
<td>S</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>A6</td>
<td>M6</td>
<td>M3</td>
<td>M1</td>
<td>A7</td>
<td>T</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Final traversal result.

<table>
<thead>
<tr>
<th>Module name</th>
<th>T</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>16</td>
</tr>
<tr>
<td>Max</td>
<td>21</td>
<td>19</td>
<td>14</td>
<td>18</td>
<td>6</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>Y/N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 1: The fault tree.

Figure 2: AND gate and OR gate preprocessing.

Figure 3: PAND gate and OR gate preprocessing.
3. BDD Subtree Calculation

The BDD method is generally applied to solve the problem of the static fault tree. The BDD method plays a great role in promoting the analysis of the static subtree. The BDD method was proposed by the American scholar Akers in 1978, as discussed in [30]. The method can transform the fault tree into a BDD only containing bottom events. The advantage of the BDD method is that the fault tree can be analyzed only using Weibull function expression without depending on intermediate events. Based on the BDD, disjoint Weibull function can be written down directly, and the minimum cut set can be obtained for qualitative analysis. The calculating amount of the method is linearly related to the scale of the graph. Therefore, it is an effective way to solve the problem that the calculating amount of the traditional FTA method increases exponentially with the scale of the fault tree.

3.1. Static Fault Tree Transformation. In the BDD conversion process, the main operation is based on the if-then-else (ite) structure of the Shannon decomposition, shown in the following equations:

\[
 f(X) = x_i f_1 + \bar{x}_i f_0, \quad (1)
\]

\[
 f_1 = f_1(x_1, x_2, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n),
 f_0 = f_0(x_1, x_2, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n), \quad (2)
\]

where the parent node is \( x_i \) and the child nodes are \( f_0 \) and \( f_1 \).

3.1.1. Validation. Simple coding and index ordering are performed using ite for a simple fault tree. The influence of different index sorting on BDD generation is analyzed.

The fault tree is shown in Figure 6. The following equations represent the relationships among events:

\[
 T = x_1 + G_1,
 G_1 = x_2 \cdot x_3,
 \quad = ite(x_2, 1, 0) \cdot ite(x_3, 1, 0), \quad (3)
\]

\[
 T = ite(x_1, 1, 0) + ite(x_2, ite(x_3, 1, 0), 0),
 = ite(x_1, 1, ite(x_2, ite(x_3, 1, 0), 0)). \quad (4)
\]

The BDD conversion graph of the fault tree is drawn by the above relationship.

3.1.2. Impact Analysis of Variable Ordering Results. At present, the optimal ordering of BDD is still under study. The mainstream BDD optimization algorithms are precise sorting algorithm and dynamic heuristic algorithm. The optimal index ranking is shown in Figure 6. When the variable order becomes index (\( x_2 \)) > index (\( x_3 \)) > index (\( x_1 \)), BDD would change (Figure 7).

Since the different ordering, the original six sides change to twelve sides. The calculation process is increased from two sets of equations to four groups, and variable ordering is especially critical when dealing with more complex models. It is shown that a good variable ordering is related to the complexity of the BDD.

3.2. BDD Probability Calculation. After sorting the variables, the path pointing to 0 or 1 can be obtained, where the point 1 represents the top event occurring and the point 0 represents not occurring.

3.2.1. Calculation of Top Event Probability. Determining all paths point to 1, the path is recorded as \( P_i \) \( (i = 1, 2, 3, \ldots, m) \), where \( m \) is the number of paths. \( P_i \) is shown in the following equation:

\[
 P_i = \prod_{j=1}^{n} i_{ij} \cdot \prod_{j=1}^{n} i_{ij}^\prime, \quad (5)
\]

Disjoint expressions of the fault tree can be represented as follows:

\[
 P(T) = P \left( \sum_{i=1}^{m} P_i \right) = \sum_{i=1}^{m} \left( \prod_{j=1}^{n} x_{ij} \right). \quad (6)
\]

The probability of occurrence of the top event can be calculated using probability formula of the mutual exclusion event as below:

\[
 P(T) = P \left( \sum_{i=1}^{m} P_i \right) = \sum_{i=1}^{m} \left( \prod_{j=1}^{n} P(x_{ij}) \right). \quad (7)
\]

The probability of failure of the bottom event can be expressed as \( P(x_{ij}) \).

3.3. A Case of Study. The above content is verified by an example. A, B, C, D, and E are bottom events (Figure 8). The probability of failure of the bottom event is shown in Table 3.

After determining the probability of the leaf node, the fault tree can be modularly divided. The fault tree is divided into three parts: M1, M2, M3, where the submodules are M1 and M2. The submodule M1 is converted into BDD, as shown in Figure 9. Combining the transfer graph with the failure probability of variables B and C, probability of occurrence of module M1 can be obtained, which is 0.28.

The probability of occurrence of module M2 is calculated based on the following BDD (Figure 9). Combined with failure probability of variables D and E, the probability of occurrence of module M2 can be obtained, which is 0.405.

After obtaining the probability of occurrence of the modules M1 and M2, probability of occurrence for M3 can be further calculated (Figure 10). The BDD model of the fault tree, defined as module M3, is a combination of M1, A3, and M2.

Obtain the probability of M1, M2, and A, and the probability of M3 can be calculated as 0.657.
Figure 4: Priority and gate preprocessing.

Figure 5: FDEP gate preprocessing.

Figure 6: The fault tree transformation diagram.

Figure 7: Reordering BDD.

Figure 8: Modular fault tree.
4. Solution of Markov Model by Dynamic Fault Tree

4.1. Markov Model Theory. Continuous-time Markov decision processes provide a very powerful mathematical framework to solve widely used decision problems, as discussed by Bartocci [31]. The process can be defined as the following random process:

\[ X(t), t \in T, \] if \( t_0 < t_1 < \ldots < t_n < t_{n+1} \in T, \]

\[ P[X(t_{n+1}) \leq x_{n+1}|X(t_0) = x_0, X(t_1) = x_1, \ldots, X(t_n) = x_n], \]

\[ = P[X(t_{n+1}) \leq x_{n+1}|X(t_n) = x_n], \]

\[ A \text{ is a random variable event, which occurs before time } t_n, \text{ and event } B \text{ is an event that occurs afterwards.} \]

4.2. Markov and Dynamic Logic Gate Transformation. In order to make the fault tree model better deal with the sequential logical relationship between the parts in the dynamic system, Dugan et al. [17] proposed the dynamic gate to represent the dynamic working relationship of the system in 1992. The Markov chain can represent the conversion relationship between system states using graphics. This graphical structure provides a visual tool to solve problems. Dynamic fault trees (DFTs) introduce four basic (dynamic) gates: the priority AND (PAND), the sequence enforcing (SEQ), the standby or spare (SPARE), and the functional dependency (FDEP), as discussed by Dugan et al. [17]. The Markov transformation process of specific dynamic logic gates will be introduced.

4.2.1. Dynamic Logic Gate Transformation. The process of converting the four common dynamic logic gates into Markov chain will be described. The four common dynamic logic gates are PAND gate, SEQ, SPARE, and FDEP, and the relevant transition diagrams are shown in Figures 11–14.

4.3. Quantitative Analysis of Markov chain. The quantitative analysis method of the Markov model generally consists of five steps: first, define the system state; second, the transition probability matrix construction; third, solve the spherical transition probability matrix; fourth, solve the differential equations; fifth, solve the fault probability state at any time.

The system state set, the fault state set, and the working state set are defined as follows: \( M = \{0, 1\} \), \( F = \{1\} \), and \( W = \{0\} \).

When defining the random process, the corresponding time point \( t \) should be set as shown below:

\[ X(t) = \begin{cases} 1, & \text{system failure}, \\ 0, & \text{system working} \end{cases} \]

(11)

Defining \( A = P_E \), where \( A \) is the transition matrix and \( E \) is equal to the identity matrix of \( P \). The equation of state is shown below:

\[ \begin{cases} P' = AP, \\ P(0) = E. \end{cases} \]

(12)

The derivative column of the state probability and the column vector of the probability derivative are, respectively, expressed as \( P \) and \( P' \). Equation (13) is C–K equation:

**Table 3: Bottom event failure probability.**

<table>
<thead>
<tr>
<th>Bottom event code</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure probability</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.15</td>
</tr>
</tbody>
</table>

**Figure 9: Model M1 and model M2.**
when the chain length is 1, the state transition process can be represented as follows:

\[
P_{ij}^{(t)} = P_{i0}^{(t)} = \frac{\lambda_{0,1}}{\lambda_{0,1} + \lambda_{0,NF}} \left( 1 - e^{-(\lambda_{0,1} + \lambda_{0,NF})t} \right),
\]

\[
= \lambda_{0,1} \left\{ \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} - \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} e^{-(\lambda_{0,1} + \lambda_{0,NF})t} \right\}.
\]

\[
\lambda_{0,1} \geq 0, \lambda_{0,NF} > 0.
\]

The state transition process of chain length 2 can be deduced by the formula of chain length 1. When the event from 0 to \( j \) requires two transfer processes, this process is defined as \( T^2 \), and \( P(T) \) is the ultimate transition probability of each edge on \( T^2 \). This process can be seen as follows:

\[
P_{ij}^{(t)} = P_{i0}^{(t)} = \frac{\lambda_{0,1}\lambda_{1,1}}{\lambda_{0,1} + \lambda_{0,NF} + \lambda_{1,2} + \lambda_{1,NF}} \left( \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} - \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} e^{-(\lambda_{0,1} + \lambda_{0,NF})t} \right)
\]

\[
= \lambda_{0,1} \left\{ \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} - \frac{1}{\lambda_{0,1} + \lambda_{0,NF}} e^{-(\lambda_{0,1} + \lambda_{0,NF})t} \right\}.
\]

The following equation represents the state transition of chain length is \( n \):

\[
P_{0n}^{(t)} = \prod_{i=1}^{n} P_{0i}^{(t)} = \prod_{i=1}^{n} \frac{\lambda_{i-1,1} \lambda_{i-1,NF}}{\lambda_{i-1,1} + \lambda_{i-1,NF}} - \sum_{i=1}^{n} \left( \lambda_{i-1,1} + \lambda_{i-1,NF} \right) \prod_{j=1}^{n} \left( -\lambda_{i-1,1} - \lambda_{i-1,NF} + \lambda_{j-1,1} + \lambda_{j-1,NF} \right) e^{-(\lambda_{i-1,1} + \lambda_{i-1,NF})t},
\]

where \( \lambda_{i-1,1} > 0, \lambda_{i-1,NF} > 0 \).

5. Fault Tree Modular Analysis

The main work of the pharmaceutical system of explosive vehicle is to manufacture explosives. The working environment of the explosive vehicle is harsh, and the failure rate is high. Through existing data, the failure probability of bottom event was obtained for the explosive vehicle.

5.1. Fault Tree Establishment and Module Division. Figure 15 shows the dynamic fault tree. It can be seen that the fault tree established includes several static and dynamic modules. The system is mainly divided into four types: flow display module, storage tank module, pump body module, and sensitizing agent system module. The storage tank module is mainly responsible for storing various liquids to ensure that the catalysts used in the preparation process are kept in a stable state. The main function of the chemical agent module is to filter to ensure that no impurities are mixed in during the configuration process. The pump body module controls the stable output of the ingredients during the preparation of explosives. The flow display module ensures that the flow output process can be observed by the outside world.

5.1.1. Preprocessing the Fault Tree. The FDEP gate exists under the OR gate of module N1. The FDEP gate transformation process is as follows: the subordinate modules of N2 are all static gates. The module M1 under N2 can be combined due to a static module. Fault tree after preprocessing is shown in Figure 16.
The fault tree can be divided into four basic modules S1, S2, S3, and S4. In module S1, there is a submodule C consisting of static logic gates, which can be divided into static module. Modules S2 and S3 are static modules. Module S4 is divided into dynamic submodule because its subtree root node is the spare parts gate.

5.2 Fault Tree Modular Calculation. According to BDD, its nodes are firstly divided. Sorting the bottom events uses a heuristic sorting method and defined the order $\text{index}(x_1) > \text{index}(x_2) > \text{index}(x_3)$. Get BDD (Figure 17).

For the transformation of the static module S2, the above method should be used. First, the heuristic algorithm is applied to determine the subunit sequencing:

$$in de x(x_1) > in de x(x_2) > in de x(x_3) > in de x(x_{10}) > in de x(x_{11}) > in de x(x_{12}) > in de x(x_{13}).$$  \hspace{1cm} (19)

The BDD of module S2 is shown in Figure 18.

Similarly, module S3 was converted into BDD, where the sorting is defined as

$$in de x(x_{15}) > in de x(x_{16}) > in de x(x_{17}) > in de x(x_{18}).$$  \hspace{1cm} (20)

The BDD of transformation is shown in Figure 19.

The module S4 contains the submodules C4 and C5, where C5 is the spare part of C4. BDD was established to solve the probability of C4 (Figure 13). The order has no succession, and Markov chain is shown in Figure 20.

The failure probability of parts in the bottom event is shown in Table 4. The following data sources are from the maintenance records of the BZ15 heavy-duty oil ammonia explosive vehicle and related experimental data in the factory.

The probability of failure of each module at 1000 hours is obtained by the bottom event probability, as shown in Table 5.

After obtaining the data failure probability of each module, the data are further integrated to obtain failure probability of the top event.
According to the integration, the reliability of the system after one thousand hours of running time can be obtained. The integration diagram is shown in Figure 21.

Finally, combining the failure probability of each module to evaluate the system reliability and failure rate can be obtained as 0.6292 and 0.3708, respectively.
6. Discussion

The Markov model is usually used to solve the problem of dynamic fault tree. Complex systems are difficult to build using traditional Markov models. Due to the dynamic fault tree containing static subtrees, the BDD can be used to solve the problem of static subtree. The logic gate combination is simplified using the pretreatment method. For the simplified dynamic fault tree [32–35], BDD is used to solve the problem. The complexity of the solution is closely related to the construction of the binary decision graph model. The complexity of model construction is affected by the ordering.
of variables. The complex structure of the fault tree is improved, and the difficulty of sorting the BDD variables is reduced by preprocessing the logic gate. Taking the subtree N1 in Figure 16 as an example, the FDEP gate is connected under the OR gate. If the variables are sorted directly without preprocessing, it is difficult to reasonably sort due to the various sorts of dynamic logic gate. The preprocessed module is shown in Figure 17. The preprocessed module S1 only contains two kinds of static gates. Sorting the variables is becoming easier, and difficulty of solving is reduced. The binary decision method can be directly used to solve the solution, and the result is as shown in Table 5. Using this method, the failure probability of each subtree can be obtained at the same time, and the dynamic fault tree can be rigorously analyzed.

The information obtained from the current research results is still very limited. Because of its influence by many factors, the accuracy of the failure probability still needs further research. Therefore, the subsequent researches should go along diversification. Since the amount of information that can be obtained using only one data source is small, in the future, a multisource information fusion

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Event</th>
<th>Probability (10⁻⁴h⁻¹)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>Line interface failure</td>
<td>0.5</td>
</tr>
<tr>
<td>X2</td>
<td>Fuel flow-meter failure</td>
<td>0.4</td>
</tr>
<tr>
<td>X3</td>
<td>Sensitizer flow-meter failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X4</td>
<td>Solenoid valve failure</td>
<td>0.3</td>
</tr>
<tr>
<td>X5</td>
<td>Connection line failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X6</td>
<td>System display failure</td>
<td>0.1</td>
</tr>
<tr>
<td>X7</td>
<td>The leakage of fuel tank</td>
<td>0.1</td>
</tr>
<tr>
<td>X8</td>
<td>The leakage of oil pipe joints</td>
<td>0.5</td>
</tr>
<tr>
<td>X9</td>
<td>Oil pipe failure</td>
<td>0.1</td>
</tr>
<tr>
<td>X10</td>
<td>Oil filter element failure</td>
<td>0.8</td>
</tr>
<tr>
<td>X11</td>
<td>Oil-pressure meter failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X12</td>
<td>One-way valve failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X13</td>
<td>Stator failure</td>
<td>0.45</td>
</tr>
<tr>
<td>X14</td>
<td>Bearing failure</td>
<td>0.15</td>
</tr>
<tr>
<td>X15</td>
<td>Emulsion matrix pump failure</td>
<td>0.3</td>
</tr>
<tr>
<td>X16</td>
<td>Flexible pipe failure</td>
<td>0.7</td>
</tr>
<tr>
<td>X17</td>
<td>Flexible pipe joint failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X18</td>
<td>Butterfly valve failure</td>
<td>0.4</td>
</tr>
<tr>
<td>X19</td>
<td>Sensitizer pump failure</td>
<td>0.1</td>
</tr>
<tr>
<td>X20</td>
<td>Water tube failure</td>
<td>0.5</td>
</tr>
<tr>
<td>X21</td>
<td>Sensitizer filter element failure</td>
<td>0.2</td>
</tr>
<tr>
<td>X22</td>
<td>Sensitizer tank failure</td>
<td>0.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Model</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Fuel flow module</td>
<td>0.1008</td>
</tr>
<tr>
<td>S1</td>
<td>Flow display module</td>
<td>0.1537</td>
</tr>
<tr>
<td>S2</td>
<td>Tank module</td>
<td>0.2261</td>
</tr>
<tr>
<td>S3</td>
<td>Pump body module</td>
<td>0.1513</td>
</tr>
<tr>
<td>S4</td>
<td>Sensitizing agent system module</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

Figure 21: Integration diagram.
method will be used to more accurately evaluate the system reliability.

7. Conclusion

This article introduces how to use the modular approach to solve the probability density of basic events in the dynamic fault tree when the system has certain complexity and multiple subsystems. The module pretreatment method is adopted to simplify the submodule of the fault tree due to the excessive mixture and overlap of logic gates. This modular approach has two advantages: first the variables ordering for the module becomes easy to reduce the complexity of BDD and solve difficulty by pretreatment, and second the failure probability of the subsystem can be obtained for evaluating the reliability of the subsystem. The feasibility of the method was verified by analyzing dynamic fault tree of explosive production system for the BCZH-15 explosive vehicle. At present, due to the difficulty of individual data collection in the data collection process and the lack of a large amount of experimental data support, the failure rate of the collected products is not accurate enough. It may have an impact on the accuracy of the result evaluation. In the future, when the data are insufficient, fuzzy theory will be used to define the failure rate of the product, and the rationality of the final output result will be determined through the comparison of expert experience.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

There are no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China under the contract no. 71761030 and the Graduate Teaching Program of Inner Mongolia University of Technology under the contract no. YJG2017013.

References


