Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3737486, 10 pages
https://doi.org/10.1155/2019/3737486

Research Article

A Test Scenario Automatic Generation Strategy for Intelligent Driving Systems

Feng Gao,[1] **Jianli Duan**,[1] **Yingdong He**,[2] **and Zilong Wang**[3]

[1]*School of Automotive Engineering, Chongqing University, Chongqing 400044, China*
[2]*Mechanical Engineering, University of Michigan, MI 48109, USA*
[3]*China Automotive Technology and Research Center, Tianjin 300300, China*

Correspondence should be addressed to Feng Gao; gaofeng1@cqu.edu.cn

Received 27 June 2018; Revised 30 September 2018; Accepted 30 December 2018; Published 15 January 2019

Academic Editor: Emilio Insfran Pelozo

In this paper, a methodology of automatic generation of test scenarios for intelligent driving systems is proposed, which is based on the combination of the test matrix (TM) and combinatorial testing (CT) methods together. With a hierarchical model of influence factors, an evaluation index for scenario complexity is designed. Then an improved CT algorithm is proposed to make a balance between test efficiency, condition coverage, and scenario complexity. This method can ensure the required combinational coverage and at the same time increase the overall complexity of generated scenarios, which is not considered by CT. Furthermore, the way to find the best compromise between efficiency and complexity and the bound of scenario number has been analyzed theoretically. To validate the effectiveness, it has been applied in the hardware-in-the-loop (HIL) test of a lane departure warning system (LDW). The results show that the proposed method can ensure required coverage with a significantly improved scenario complexity, and the generated test scenario can find system defects more efficiently.

1. Introduction

In recent years, with the development of intelligent driving technologies, such advanced driver assistance systems (ADAS) as auto emergency braking (AEB), forward collision warning (FCW), lane departure warning (LDW), etc. have been put into market rapidly [1]. Compared with the traditional onboard electronic systems, the working condition of such systems is uncontrolled and indefinable exactly, which causes the traditional test scenario design approach to be inapplicable. On the other hand, since these systems influence the driving safety directly, sufficient and comprehensive tests are necessary before they go public [2, 3].

At this stage, the approaches of designing the test scenario for ADAS can be divided into the following five types mainly: naturalistic field operational test (N-FOT), Monte Carlo simulation (MCS), accelerated evaluation method (AE), worst-case scenario evaluation (WCSE), and test matrix approach (TM). When using N-FOT, vehicles equipped with ADAS to be tested are driven by multiple drivers in real traffic for data collection over a quite long period to achieve enough coverage [4]. It can test the system under the real working condition, but the occurrence probability of critical condition is very low and most of the time the system may be tested under the similar and simple scenarios. Moreover, the real traffic is uncontrollable, which makes it difficult to ensure a good test consistency [5]. To improve N-FOT, some researchers use the data collected from naturalistic traffic to build a stochastic driving behavior model to generate test scenarios, which is called MCS. Based on this approach, Yang et al. developed a car-following driver model to evaluate FCW and AEB [6]. Compared with N-FOT, MCS can partly extend the test condition by the driving behavior model, ensure a good consistency with real traffic, and reduce the similar and simple scenarios to increase test efficiency. Unfortunately, the critical condition is still hardly to be generated because of the lack of original traffic data.

To generate more critical scenarios, Zhao et al. [7] and Huang et al. [8] proposed AE, which applies importance-sampling theory to speed up the evaluation process by finding the most critical scenarios. However, this method only takes limited influence factors into account for some specific

driving assistance function, which restricts its application range. In order to directly extract the most critical scenarios, WCSE was proposed based on the database built up from the traffic accidents. Andreas validated the emergency lane assist system based on these most dangerous scenes [9]. These extreme scenarios help to find the faults and achievable performance of ADAS, but the coverage of all related factors cannot be measured quantitatively. This makes it difficult to determine when to stop the validation and evaluation process.

As a widely used method to construct the standardized test scenarios, TM has also been applied in the formulation of test standards for ADAS, e.g., ISO 15622 for ACC [10], AEB in EuroNCAP [11], etc. TM can take all factors relating to the tested system into consideration. Therefore compared with other methods, it can be used to design higher diversity and complex scenarios with an acceptable testing cost, better controllability, and preferable repeatability. However, the current TM has the following problems when applied on ADAS test [12]: (1) the number of test scenarios and considered factors is too few to achieve the coverage requirement. Otherwise the mostly used orthogonal experiment method (OE) for TM leads to "dimensional disaster" when the number of considered influence factors increases; (2) and its test efficiency is very low when failures are only triggered by the interaction of a small number of factors.

To solve these issues of TM, we introduce the combinatorial testing method (CT), which has already been widely used in computer software testing [13]. The motivation of CT is based on the fact that most of the program errors are caused by the interaction of multiple influence factors [14]. The set of test scenarios generated by CT can guarantee full coverage of the required n-wise combination with as few scenario number as possible. Cohen et al. proposed Automatic Efficient Test Generator (AETG) algorithm, which generates a certain number of test scenarios at each time by using the greedy algorithm and then picks one that can cover most of the n-wise combination of factors still uncovered [15]. Czerwonka developed another CT tool called Pairwise Independent Combinatorial Testing tool (PICT), which is unlike AETG algorithm and need not produce test scenarios to be selected beforehand [16]. Instead it uses a fixed random seed to ensure the consistency of the output. To trade off a reduction of the number of test scenarios, James developed a new tool called Allpairs to find a reasonably small set of test scenarios to satisfy the full coverage of arbitrary pairwise combination [17]. Wu et al. used the particle swarm optimization algorithm to generate test suites ensuring the coverage criteria with smaller test set than other metaheuristic strategies [18]. Garvin et al. modified the original simulated annealing algorithm to further improve the efficiency of test suite generation [19]. Gonzalez-Hernandez used the Mixed-Tabu Search to build the test suite with uniform strength [20].

All the aforementioned CT methods focus on the reduction of test set while ensuring the combinatorial coverage at the same time. The complexity of scenario has not been taken into account when generating combinatorial test scenarios, which is beneficial to test effectiveness because a system tends to malfunction under more complex conditions. Therefore, a new method called combinatorial testing based on scenario complexity (CTBC) is presented in this paper for automatic generating the test scenarios for intelligent driving system, which can achieve full coverage of n-wise combination with a relative compact test set meanwhile increasing the scenario complexity. The following paper is organized as follows: Section 2 describes the model of influence factors and potential problems; Section 3 introduces the procedure and principle of the proposed CTBC, whose performance is analyzed theoretically in Section 4; Section 5 validates CTBC by application and comparative analysis; and finally Section 6 concludes the paper.

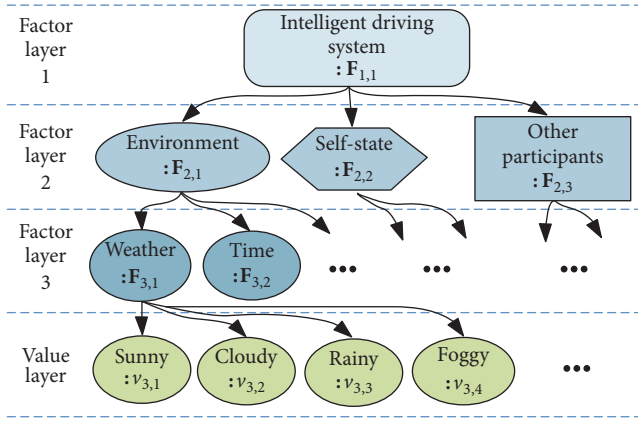

Figure 1: Tree structure model of influence factors.

2. Problem Description

Before discussing the problem of automatic generation of test scenario for intelligent driving systems, a mathematical model is introduced to describe the influence factors of intelligent driving system, which act as the input of the automatic generation process of test scenarios.

2.1. Tree Structure Model of Influence Factors. The components used to construct the test scenario are called influence factors in this paper, which are related to the functionality of intelligent driving system. These factors can be obtained through a variety ways, such as technical specifications, naturalistic traffic data, etc. To ensure the comprehensiveness, such systematic methods, e.g., classification tree [21], are suggested to analyze the possible influence factors. In general, the influence factors of an intelligent driving system can be divided into three types: environment, self-state, and other road participants.

To realize automatic generation of test scenarios, the influence factors should be discretized to get distinct values, which can be generated by such black box test scenarios design methods as equivalence class partition and boundary value processing [12]. Finally, a tree structure model of influence factors can be derived as shown in Figure 1 as an example. Table 3 in the Appendix shows the detailed influence factors of LDW.

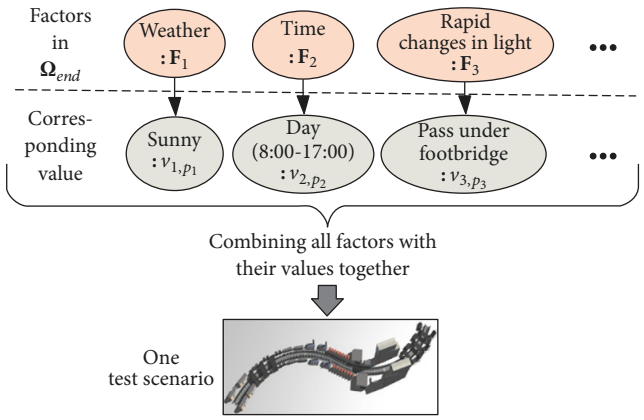

Figure 2: Test scenario construction process.

Then the factors and their values can be described by a general mathematical linguistic form as shown by (1) and (2):

$$\mathbf{F}_{i,j} = \left\{\mathbf{F}_{i+1,k}, k \in \mathbf{\Omega}_{i,j}\right\} \quad i \in [1, \cdots, S-1], \ j \in [1, \cdots, L_i] \tag{1}$$

where $\mathbf{F}_{i,j}$ is the j-th factor at the i-th layer, $\mathbf{\Omega}_{i,j}$ is composed of all subfactors of $\mathbf{F}_{i,j}$, S is the number of factor layer, and L_i is the number of factors in the i-th layer.

All subfactors belonging to the same parent factor can appear in one scenario, while the different values of the same factor at the bottom layer cannot. Therefore, we need a different formula to describe all possible values of factors locating at the bottom factor layer:

$$\mathbf{F}_{i,j} = \left\{v_{i,k}, k \in \mathbf{\Omega}_{i,j}\right\}, \quad (i,j) \in \mathbf{\Omega}_{end}, \tag{2}$$

where $v_{i,k}$ is the k-th value of $\mathbf{F}_{i,j}$ and $\mathbf{\Omega}_{end}$ is composed of the subscripts of all factors at the bottom layer.

Finally, the test scenario can be built by combining all the factors at the bottom factor layer with one of their corresponding values together. To simplify the description, let $|\bullet|$ denote the number of elements belonging to a set. Then the end node factors can be denoted as $\mathbf{F}_q(q = 1, \cdots, N)$, where $N = |\mathbf{\Omega}_{end}|$. And each factor's values can be denoted as $v_{q,p_q}(p_q = 1, \cdots, |\mathbf{F}_q|)$. The mathematical expression of test scenarios is

$$\mathbf{T} = \left[t_{i,j}\right] \in \mathbb{R}^{M \times N}, \quad t_{i,j} \in \left\{v_{j,1}, \cdots, v_{j,|\mathbf{F}_j|}, \ i = 1, \cdots, M, \ j = 1, \cdots, N, \right. \tag{3}$$

where $\mathbf{T}$ is a matrix including all test scenarios, M is the number of test scenario, and $t_{i,j}$ denotes the value of j-th factor in the i-th scenario. The construction process of a test scenario is shown in Figure 2 as an example.

2.2. Problems Analysis. Based on the structure model of influence factors of LDW shown in Table 3 in the Appendix, the problem of generation of effective test scenario is discussed in this section. There are 16 influence factors at the bottom level, which have 58 values totally. The number of test scenarios defined in ISO 17361 is only 8 [22], which is far away to cover all possibilities and its error detection capability is pretty bad. Such standard test scenarios only can be used to validate the primary functionality. On the other hand, the scenarios generated by OE can ensure the coverage of all influence factors, but the number of scenarios reaches 349,920,000, which is unacceptable in practice because of the test cost. By using the PICT, the number of scenarios ensuring pairwise combination is reduced to 53 [23]. However, the percentage of scenarios with higher complexity cannot be improved, under which LDW may malfunction easily. Therefore in order to increase the proportion of scenario with higher complexity to further improve the test efficiency, an improved CT method called CTBC is to be introduced in the next section.

3. Combinatorial Test Generation Method Based on Complexity

To control the complexity of the generated test scenarios, we need an index to measure the contribution of factor value to the complexity of scenario.

3.1. Complexity Index of Scenario. The contribution of each factor or value to the complexity of scenario can be determined by the Analytic Hierarchy Process (AHP), which is widely used in the field of engineering because of its ability to make quantitative analysis of subjective evaluation [24]. The calculation process is based on the tree structure model of influence factors as shown in Figure 1. According to their relative contribution, a judgment matrix $\mathbf{A}_{i,j}$ can be constructed as

$$\mathbf{A}_{i,j} = \left[a_{h,f}\right] \in \mathbb{R}^{|\mathbf{\Omega}_{i,j}| \times |\mathbf{\Omega}_{i,j}|} \quad i \in [1, \cdots, S-1], \ j \in [1, \cdots, L_i], \ h, f = 1, \cdots, \left|\mathbf{\Omega}_{i,j}\right| \tag{4}$$

where $a_{h,f}$ represents a comparison of the relative importance of any two elements in $\mathbf{\Omega}_{i,j}$, and it can be determined by Saaty's scaling law [24]. It also has the following properties:

$$a_{h,f} = \begin{cases} 1 & h = f \\ \dfrac{1}{a_{f,h}} & h \neq f \end{cases} \tag{5}$$

Then, the normalized eigenvector corresponding to the maximum positive eigenvalue of $\mathbf{A}_{i,j}$ is

$$\mathbf{W}_{i,j} = [\omega_{i,k}] \in \mathbb{R}^{1 \times |\mathbf{\Omega}_{i,j}|}, \quad k \in \mathbf{\Omega}_{i,j} \tag{6}$$

where $\omega_{i,k}$ represents the relative importance of influence factors or its corresponding values according to [24].

However, the relative importance degree is not comparable for the factors belonging to different categories. Therefore to measure the importance of different values used to construct test scenario uniformly and equally, the following importance index of value is used:

$$\widehat{\omega}_{i,k} = \prod_{(a,b) \in \widehat{\mathbf{\Omega}}_{i,k}} \omega_{a,b}, \quad k \in \mathbf{\Omega}_{i,j}, \ (i,j) \in \mathbf{\Omega}_{end} \tag{7}$$

Table 1: Variables used in pseudocode of CTBC.

Variables	Meanings
threshold	The threshold value
β	The complexity improvement coefficient, $\beta \in [0\%, 100\%]$
uncovered	The set of combinations that remained uncovered
comb	The combination in *uncovered*
combweight	The sum of the values' importance indices
setofweight	The set of *combweight*
testscenario	The new test scenario
best	The first chosen combination with max(*combweight*) in current *uncovered* in *testscenario*
nextbest	The combination with max(*combweight*) in current *uncovered*

where $\widehat{\mathbf{\Omega}}_{i,k}$ is composed of the subscript of the nodes in the route from $v_{i,k}$ to the root node $\mathbf{F}_{1,1}$. Table 3 shows the importance index of factor values of LDW. Then, with the importance index, the complexity index $\mathbb{C}_i$ of the i-th test scenario in $\mathbf{T}$ is defined as

$$\mathbb{C}_i = \sum_{j=1}^{N} \widehat{\omega}_{i,j}, \tag{8}$$

where $\widehat{\omega}_{i,j}$ is the importance index of $t_{i,j}$.

3.2. Complex Index Based CT Algorithm. In this section, a CTBC algorithm is introduced to improve the test scenario complexity while ensuring required combinational coverage. An idea to improve the scenario complexity is that the factor value with higher importance index is preferred to generate the test scenario. Naturally more complex scenarios can be found at the beginning of automatic generation process. But the importance index of the left factor value not covered becomes very small. It is impossible to construct complex scenario using these factor values, which is bad for the overall complexity of all generated scenarios.

To overcome this problem, a threshold is defined to choose the aspect to be considered when generating test scenarios. When generating a new scenario, if the summed importance index of the selected factor values is larger than this coefficient, the unselected factor values with the most combinational coverage is preferred. Otherwise, the importance index is considered preferentially when determining the value of left factors of a scenario. By this way, a tradeoff between combinational coverage and scenario complexity is made.

For a given tree structure model of influence factors with their importance index (4), there exists a boundary of achievable maximum or minimum complexity of scenario. If the selected threshold is less than the minimum complexity, the CTBC algorithm generates scenarios only considering the combinational coverage requirement like AETG. In Section 4.1, a complexity improvement coefficient is proposed to determine a proper threshold.

The CTBC algorithm generates one test scenario at a time and lasts until the combinations of all possible values have been covered by at least one scenario. Based on the aforementioned idea and the variables defined in Table 1, the pseudocode of CTBC is shown in Algorithm 1.

When programming the executable code, the following should be noted:

(a) In order to accelerate the searching process, all uncovered combinations are put in a red-black tree according to the sum of importance indices in the descending order [25].

(b) Moreover, the requirement of reproducibility and certainty is important for the test of ADAS. Therefore the lexicographical order is used to ensure the consistency of generated scenarios when there exist multiple choices providing the same sum of importance indices [26].

4. Performance Analysis of CTBC

The generated test scenarios are evaluated mainly by two aspects: (a) test cost measured by the number of scenarios (b) and test effectiveness evaluated by the overall scenario complexity in this paper. To make an optimal tradeoff between these two aspects, a complexity improvement coefficient is proposed, by which the best test scenarios can be found with a statistical method.

4.1. Optimization by Complexity Improvement Coefficient. The purpose of introducing this complexity improvement coefficient β is to improve the overall scenario complexity:

$$\beta = \frac{threshold - \underline{\mathbb{C}}}{\overline{\mathbb{C}} - \underline{\mathbb{C}}} \times 100\% \tag{9}$$

where $\underline{\mathbb{C}}$ and $\overline{\mathbb{C}}$ are the minimum and maximum achievable scenario complex indexes. From the fundamental of CTBC, a more complex scenario can be obtained by increasing β, but the number of needed scenarios ensuring combinational coverage may also increase. To make a balance between the test cost and effectiveness to find the best test scenarios, the overall complexity of test scenarios is defined firstly:

$$\widetilde{\mathbb{C}}(\beta) = \frac{\sum_{i=1}^{M(\beta)} \mathbb{C}_i(\beta)}{M(\beta)} \tag{10}$$

where $\widetilde{\mathbb{C}}(\beta)$, $M(\beta)$, and $\mathbb{C}_i(\beta)$ are the overall complexity, the number of test scenarios, and the i-th scenario's complexity

1: **Input:** $\mathbf{F}_q(q = 1, \cdots, N)$, $v_{q,p_q}(p_q = 1, \cdots, |\mathbf{F}_q|)$, $\widehat{\omega}_{q,p_q}$, n, β
2: **Output:** **T**
3: Obtain *uncovered*
4: **for** all *comb* **in** *uncovered* **do**:
5: $combweight = \sum_{\widehat{\omega}_{q,p_q} \in comb} \widehat{\omega}_{q,p_q}$
6: Add *combweight* to *setofweight*
7: **end for**
8: $threshold = \beta \sum_{q=1}^{N} \max(\widehat{\omega}_{q,p_q})$
9: **while** *uncovered* $\neq \emptyset$ **do**
10: *best* = combination with max(*combweight*) in *uncovered*
11: **for all** i in n **do**
12: Assign factors and values from combination that corresponding to *best* to *testscenario*
13: Remove *best* from *uncovered*
14: **end for**
15: **if** *best* > *threshold* **then**
16: **while** ((length of *testscenario*) < N) **do**
17: *nextbest* = combination with max(*combweight*) in *uncovered*
18: Comparing values of the same factors between *nextbest* and *testscenario*
19: **if** there exist conflicting values **then**
20: continue to select the next combination corresponding to *nextbest* in the descending order of *combweight*
21: **end if**
22: Factors and values that corresponding to *nextbest* but not exist in *testscenario* are assigned to *testscenario*
23: Remove *nextbest* from *uncovered*
24: **end while**
25: **else**
26: Assign factor $\mathbf{F}_q$ that are not contained in *testscenario* with the value of $\max(\widehat{\omega}_{q,p_q})$
27: **end if**
28: Add *testscenario* to **T**
29: **end while**

Algorithm 1: Pseudocode for CTBC.

index, which all are functions of the complexity improvement coefficient β.

Here the AHP method is used again to determine the weighting of test cost and effectiveness [24]. Then the optimization problem can be described by

$$\max_{\beta} Z,$$

$$Z = S_1 \widetilde{\mathbb{C}^*} - S_2 M^*,$$

$$\widetilde{\mathbb{C}^*} = \frac{\widetilde{\mathbb{C}}(\beta) - \underline{\mathbb{C}}}{\overline{\mathbb{C}} - \underline{\mathbb{C}}}, \tag{11}$$

$$M^* = \frac{M(\beta) - \underline{M}}{\overline{M} - \underline{M}}$$

where Z denotes the composite test effect, S_1 and S_2 are the normalized weight for scenario complexity and test cost determined by AHP method, $\widetilde{\mathbb{C}^*}$ and M^* are the normalized value of complexity and test cost, and $\overline{M}$ and $\underline{M}$ are the maximum and minimum number of scenarios. In (8), the "min-max scaling" process is used to put $\widetilde{\mathbb{C}}(\beta)$ and $M(\beta)$ in the same scale to avoid the challenge of selecting a proper weight.

For a given tree structure model of influence factors and its importance index values, the achievable maximum or minimum complex index of scenario can be derived by selecting the factor value with highest or lowest importance index. Unfortunately the range of scenario number cannot be obtained from the previously known information of the tested system, which is used to normalize the test cost in (8). In the following section, an approximated method is presented to estimate the range of scenario number.

4.2. Estimation of Test Scenario Number. The test scenario number in **T** can be discussed in two situations: (a) **T** is a Covering Array (CA) and (b) **T** is a Mixed Covering Array (MCA). The difference between CA and MCA is whether the number of each factor value is the same, i.e., in CA $|\mathbf{F_1}|=|\mathbf{F_2}|=\cdots=|\mathbf{F_N}|=u$, and in MCA the number of values is different [27].

Firstly, the simpler condition is considered, i.e., **T** is a CA. At this condition, if each factor value is only required to appear at least once, then u test scenarios are needed. If all combinations of values for any n factors are required to be covered, the lower bound of scenario can be found by the orthogonal table [28]. And Bush proposed a construction method of the orthogonal table for CA, which has the smallest

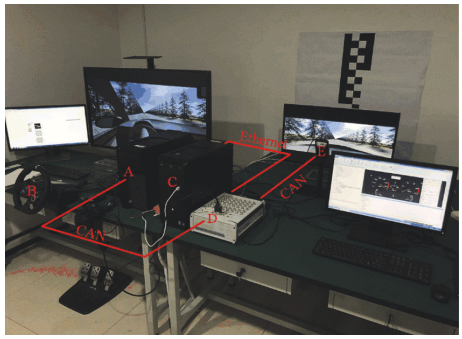

Figure 3: Hardware-in-loop test system.

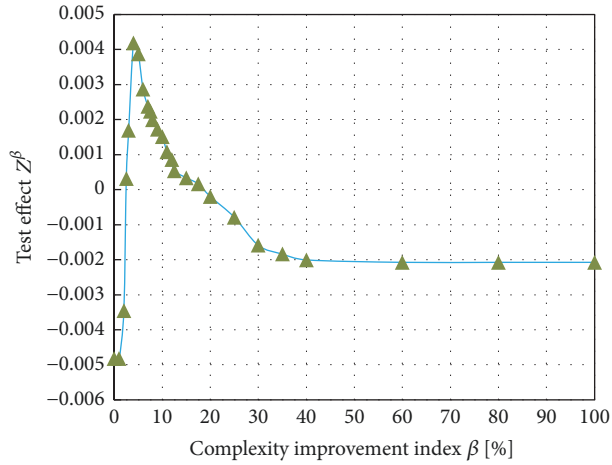

Figure 4: Optimization process of test effect.

number of scenarios [29]. At the condition $\beta = 0$, CTBC algorithm gives priority to test cost and the number of its generated scenario is no less than that of the orthogonal table [29]:

$$\underline{M} = u^n \tag{12}$$

The maximum number of scenarios is reached when $\beta = 100\%$. This means that when a new test scenario is generated, it can only cover one combination in the set of uncovered combinations and the upper bound of scenario number can be estimated by

$$\overline{M} = \binom{N}{n} u^n = \frac{N! u^n}{(n!\,(N-n)!)} \tag{13}$$

where $\binom{N}{n}$ denotes the number of options for selecting any $n (n \le N)$ factors from N ones.

Based on the aforementioned analysis, the condition that $\mathbf{T}$ is a MCA is discussed next. Firstly all factors are rearranged according to the number of their values from large to small, and we get $\mathbf{F}_1^*, \mathbf{F}_2^*, \cdots, \mathbf{F}_N^*$ satisfying $|\mathbf{F}_1^*| \ge |\mathbf{F}_2^*| \ge \ldots \ge |\mathbf{F}_N^*|$. Similar to CA, the lower bound of scenario number may happen when $\beta = 0$ and can be estimated by

$$\underline{M} = \left|\mathbf{F}_1^*\right|^n \tag{14}$$

Analogously, when $\beta = 100\%$ the maximum number of scenarios may be reached, which is calculated by

$$\overline{M} = \sum_{\mathbf{C}_i \in \mathbf{C}(N,n)} \prod_{j \in \mathbf{C}_i} \left|\mathbf{F}_j\right| \tag{15}$$

where $\mathbf{C}(N, n)$ denotes the set of combinations for selecting any $n (n \le N)$ factors from N factors, and $\mathbf{C}_i$ is the i-th combination of factors in $\mathbf{C}(N, n)$.

5. Application and Analysis

In this section, the proposed test scenario generation method is applied to evaluate an LDW system to validate its effectiveness by hardware-in-the-loop test as shown in Figure 3.

In Figure 3, "A" is a workstation where the virtual reality simulation software "Prescan" runs, "B" is the external driver input device, "C" is a host computer that can configure and monitor the real-time simulator, "D" is the MicroLabBox produced by Dspace and acting as a real-time simulator to run the vehicle dynamical model, and "E" is the tested LDW system.

5.1. Optimization of Complexity Improvement Coefficient. With this application, the strength of combination coverage is selected to be $n = 2$ to find the optimal complexity improvement index. The judgment matrix $\mathbf{A} = \begin{bmatrix} 1 & 6 \\ 1/6 & 1 \end{bmatrix}$ is used to determine the normalized weights of scenario complexity and test cost:

$$\mathbf{S} = \begin{bmatrix} S_1 & S_2 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} 0.8333 & 0.1667 \end{bmatrix}^{\mathrm{T}}. \tag{16}$$

The optimization process of test effect $Z(\beta)$ is shown in Figure 4. The best test effect is 0.0042 when $\beta = 4\%$. At this condition, $M(\beta) = 324$ and $\widetilde{\mathbb{C}}(\beta) = 0.4769$. A tradeoff between the test cost and the scenario complexity is successfully made by the proposed approach to achieve better test effectiveness.

5.2. Fundamental Validation of CTBC. The fundamental of the CTBC algorithm is that the ADAS under test is easier to malfunction under more complex scenarios. The scenarios are generated by using $n = 2$ and $\beta = 4\%$, which is optimized in Section 5.1. The number of scenarios generated by CTBC is 324, among which 10 scenarios, whose complex index distributes uniformly, are selected to test the LDW. These scenarios are numbered from 1 to 10 in descending order according to their complex index. The fault detection rate under different scenarios is shown in Figure 5.

As can be seen from Figure 5, scenario No. 1 has a fault detection rate of 42.20%, while that of scenario No. 10 is only 5.9%. The fault detection rate fluctuates between 35.65% and 43.80% as the complexity index falls from 0.4729 of scenario No. 1 to 0.4334 of scenario No. 7. And the fault detection rate reduces dramatically from scenario No. 7 to No. 10. The results show that overall the bigger the complex index of scenario is, the easier it is to find the malfunctions of tested system. This fact can be used not only to guide the automatic

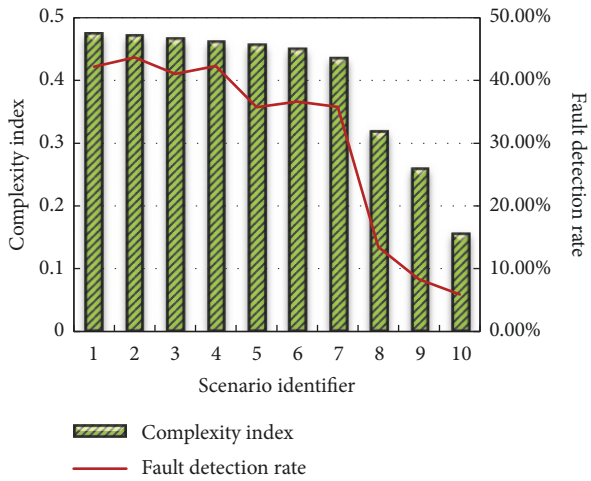

Figure 5: Relationship between complexity and fault detection rate.

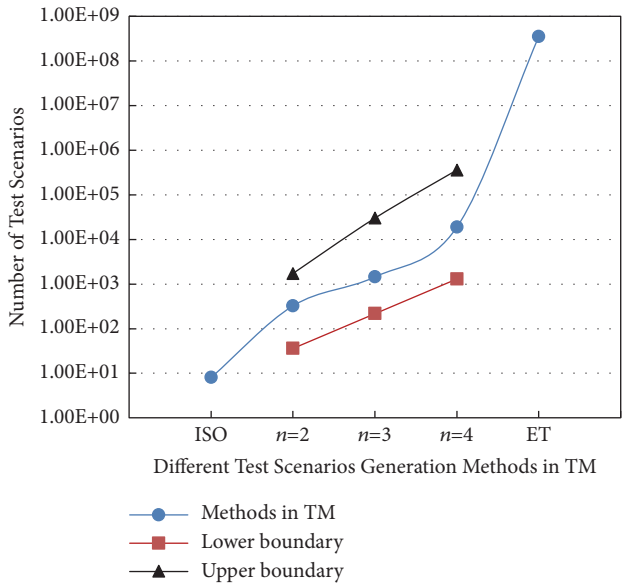

Figure 6: Number of test scenarios designed by different TM methods.

generation algorithm of test scenario, but also to evaluate the effectiveness of test scenarios generated by other algorithms.

5.3. Performance Analysis. In this section, CTBC algorithm is compared with other TM methods to validate its improvement of test effect. The number of test scenarios defined in ISO 17361, designed by the OE method, and generated by CTBC algorithm with the strength of combination coverage $n = 2, 3, 4$ is shown in Figure 6. Being similar to the optimization process in Section 5.1, the best complexity improvement coefficient is selected to be $\beta = 3\%$ when $n = 3, 4$.

It can be seen clearly that the number of the test scenario defined in ISO 17361 is too few to test the LDW adequately and throughly as there are only 8 scenarios. Moreover, these scenarios are quite simple. The number of scenarios generated by OE far exceeds others, which is almost impossible to undertake because of test costs. The scenarios generated by

Table 2: complexity index distribution.

	PICT	Allpairs	AETG	CTBC
Min	0.1360	0.0993	0.1094	0.1109
Lower quartile	0.2115	0.2142	0.1439	0.4509
Median	0.2546	0.2475	0.1935	0.4769
Upper quartile	0.2997	0.3101	0.2760	0.4884
Max	0.3699	0.4128	0.4190	0.5071

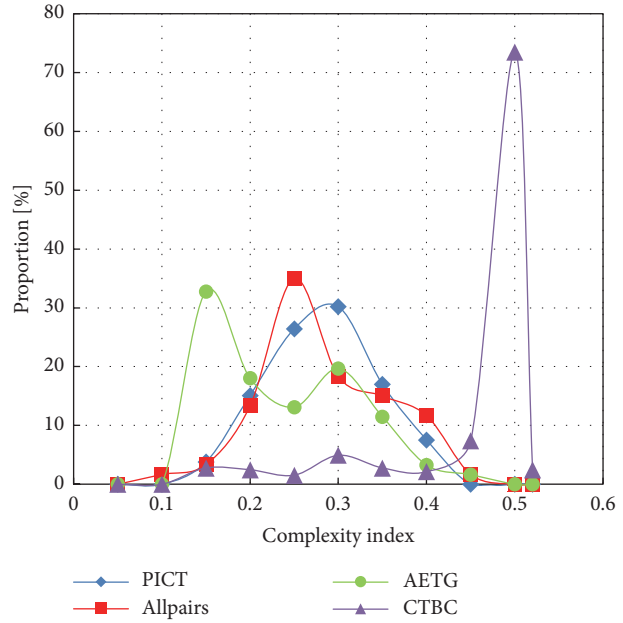

Figure 7: Comparison of the complexity index of test scenario.

CTBC are more reasonable by synthetically considering the test effectiveness. Furthermore from the previous studies it is known that a quite small n can reveal most of potential errors [26].

Moreover the bound of scenario number is the basis of the optimization process of test scenario introduced in Section 4.1. From the results in Figure 6, it is found that the number of scenarios generated by CTBC lies in the estimated range under different coverage strength, which implies that the proposed estimation of bound of scenario number in Section 4.2 is effective.

5.4. Scenario Complexity Index Improvement. The best advantage of CTBC algorithm is its improvement of scenario complexity index, which is beneficial to finding fault validated in Section 5.2. The primary focus of this experiment is to set comparison with the test set generated by other CT algorithms which does not consider the complexity of scenario and is the basis of CTBC. Therefore, choosing such methods can highlight the superiority of the proposed algorithm. Moreover, PICT, Allpairs, and AETG have already been widely applied in engineering applications. Therefore, they are used to validate the effectiveness of CTBC in increasing the overall complexity index of test scenarios. The comparative results are shown in Figure 7 and Table 2.

Table 3: Influence factors of LDW system.

Influence factor			Value	Importance index
Traffic environment parameters	Lightning environments	Weather	Sunny	0.0126
			Cloudy	0.0475
			Rainy	0.1101
			Foggy	0.1419
		Time	Day (8:00-17:00)	0.0101
			Dusk/Dawn (17:00-19:30/5:30-8:00)	0.0403
			Night (19:30-5:30)	0.0807
		Rapid changes in light	Pass through tunnel	0.0098
			Pass under footbridge	0.0162
			None	0.0015
Traffic environment parameters	Lane lines parameters	Lane line clarity	Few fade and holes	0.0023
			Intermediate fade and holes	0.0088
			Much fade and holes	0.026
		Lane line integrity	No fade and holes	0.0009
			Lane line on one side	0.0038
			Lane line on both sides	0.0268
		Lane line number	Single	0.0148
			double	0.0049
		Lane line color	White	0.0007
			Yellow	0.0033
		Lane line type	Dashed	0.0276
			Solid	0.003
Traffic environment parameters	Road parameters	Curvature	Straight road (0)	0.0039
			Bend road (1/750)	0.0083
			Bend road (1/500)	0.0186
			Bend road (1/250)	0.0401
		Lane number	1	0.0043
			2	0.001
			3	0.0007
		Lane marks	One mark	0.002
			Combination of two marks	0.0039
			Combination of three marks	0.008
			Combination of four marks	0.0157
		Slope	None	0.001
			Uphill (Slope of +5%)	0.0068
			Downhill (Slope of -5%)	0.0044
		Roadside facilities	None	0.0011
			One facility	0.0062
			Combination of two facilities	0.0103
			Combination of three facilities	0.0176
			Combination of four facilities	0.0277
			Combination of five facilities	0.052
			None	0.0038
Subject vehicle driver's behavior	Longitudinal speed		40 km/h	0.0016
			55 km/h	0.018
			60 km/h	0.0148
			80 km/h	0.0148
			90 km/h	0.0111
			100 km/h	0.0094
			120 km/h	0.0072
			140 km/h	0.0014

Table 3: Continued.

Influence factor		Value	Importance index
Subject vehicle driver's behavior	Lateral speed	0 m/s	0.0013
		0.1 m/s	0.0071
		0.4 m/s	0.0092
		0.8 m/s	0.0114
		1.0 m/s	0.0102
Other traffic participants' state	Road congestion	0 lane lines missing	0.0016
		25% lane lines missing	0.0043
		50% lane lines missing	0.009
		75% lane lines missing	0.015
		90% lane lines missing	0.0215

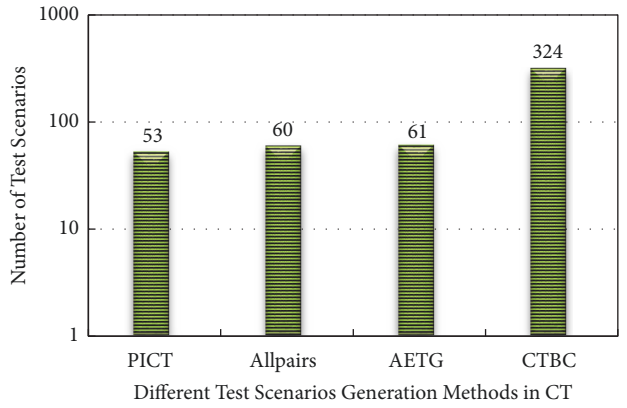

Figure 8: Comparison of test scenario number.

It can be seen that compared with other CT algorithms, though the maximum value of complexity index is only increased slightly, the number of scenarios with higher complexity index is increased significantly, which implies that the scenarios generated by CTBC can find more errors.

The number of test scenario generated by these methods is compared in Figure 8. The number of scenarios generated by CTBC is increased by 6.1132, 5.4 and 5.3115 times, respectively. It is still greatly reduced compared with that of ET (as shown in Figure 6). Besides, an appropriate increase of test cost is acceptable in order to give priority to detect the hidden errors of ADAS systems, which have respect with the driving security.

6. Conclusions

Considering the high security of intelligent driving systems, this paper proposes a new approach to automatically generate more effective test scenarios to improve the test effectiveness. The theoretical analysis and application results show the following:

(1) The larger the complexity index of scenario is, the easier it is to find out the malfunctions of tested systems.

(2) The proposed CTBC algorithm can generate more complex scenarios compared with the traditional CT method, while ensuring the required combinational coverage.

(3) Compared to the widely used ET method, the number of test scenarios generated by CTBC algorithm is reduced greatly, which leads to a dramatically cut down of test cost.

Appendix

See Table 3.

Data Availability

The tree structure model of LDW and the importance index of each factor are listed in the appendix. The ATEG algorithm is an online program, which can be found at http://alarcostest.esi.uclm.es/CombTestWeb/combinat orial .jsp. The Allpairs and PICT algorithms are free software, which are available from the corresponding author upon request. The program of CTBC algorithm programed by Python and its input file are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Key R&D Program of China under grants 2016YFB0101104 and 2017YFB0102504, Scientific Technological Plans of Chongqing under grant cstc2017zdcy-zdzx0042, and Industrial Base Enhancement Project under grant 2016ZXFB06002.

References

[1] R. A. Auckland, W. J. Manning, O. M. J. Carsten, and A. H. Jamson, "Advanced driver assistance systems: Objective and subjective performance evaluation," *Vehicle System Dynamics*, vol. 46, no. 1, pp. 883–897, 2008.

[2] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: review

and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[3] Q. Xia, J. Duan, F. Gao, T. Chen, and C. Yang, "Automatic Generation Method of Test Scenario for ADAS Based on Complexity," in *Proceedings of the Intelligent and Connected Vehicles Symposium*, Kunshan, China, 2017.

[4] D. LeBlanc, J. Sayer, C. Winkler et al., *Road departure crash warning system field operational test: methodology and results. Volume 2: appendices*, University of Michigan Ann Arbor Transportation Research Institute, 2006.

[5] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, "An overview of the 100-car naturalistic study and findings," in *Proceedings of the International Technical Conference on the Enhanced Safety of Vehicles*, p. 787, 2005.

[6] H.-H. Yang, H. Peng, T. J. Gordon, and D. Leblanc, "Development and validation of an errorable car-following driver model," in *Proceedings of the 2008 American Control Conference, ACC*, pp. 3927–3932, June 2008.

[7] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. Leblanc, "Accelerated Evaluation of Automated Vehicles in Car-Following Maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 733–744, 2018.

[8] Z. Huang, H. Lam, D. J. LeBlanc, and D. Zhao, "Accelerated Evaluation of Automated Vehicles Using Piecewise Mixture Models," *IEEE Transactions on Intelligent Transportation Systems*, no. 99, pp. 1–11, 2017.

[9] A. Eidehall, "Tracking and threat assessment for automotive collision avoidance," *Behavior*, 2007.

[10] Intelligent transport systems - adaptive cruise control systems - performance requirements and test procedures, ISO Standard 15622, 2010.

[11] European new car assessment programme - test protocol - AEB systems, NCAP Standard, 2015.

[12] Q. Zhang, D. Chen, Y. Li, and K. Li, "Research on Performance Test Method of Lane Departure Warning System with PreScan," in *Proceedings of the SAE-China Congress 2014: Selected Papers*, Lecture Notes in Electrical Engineering, pp. 445–453, Beijing, China, 2014.

[13] B. S. Ahmed, L. M. Gambardella, W. Afzal, and K. Z. Zamli, "Handling constraints in combinatorial interaction testing in the presence of multi objective particle swarm and multithreading," *Information and Software Technology*, vol. 86, pp. 20–36, 2017.

[14] G. Seroussi and N. H. Bshouty, "Vector sets for exhaustive testing of logic circuits," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 34, no. 3, pp. 513–522, 1988.

[15] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, 1997.

[16] J. Czerwonka, "Pairwise testing in real world," in *Proceedings of the 24th Pacific Northwest Software Quality Conference*, pp. 419–430, Portland, Ore, USA, 2006.

[17] Satisfice Inc., "Epistemology for the rest of us," 2008, http://www.satisfice.com/tools.shtml.

[18] H. Wu, C. Nie, F.-C. Kuo, H. Leung, and C. J. Colbourn, "A Discrete Particle Swarm Optimization for Covering Array Generation," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 575–591, 2015.

[19] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102, 2011.

[20] L. Gonzalez-Hernandez, "New bounds for mixed covering arrays in t-way testing with uniform strength," *Information and Software Technology*, vol. 59, pp. 17–32, 2015.

[21] R. D. Ona and J. D. Ona, "Analysis of transit quality of service through segmentation and classification tree techniques," *Transportmetrica A: Transport Science*, vol. 11, no. 5, pp. 365–387, 2015.

[22] Intelligent transport systems – lane departure warning systems - performance requirements and test procedures, ISO Standard 17361, 2017.

[23] R. Kuhn, Y. Lei, and R. Kacker, "Practical combinatorial testing: beyond pairwise," *IT Professional*, vol. 10, no. 3, pp. 19–23, 2008.

[24] R. W. Saaty, "The analytic hierarchy process-what it is and how it is used," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 9, no. 3-5, pp. 161–176, 1987.

[25] A. Mantler and H. Cameron, "Constructing red-black tree shapes," *International Journal of Foundations of Computer Science*, vol. 13, no. 6, pp. 837–863, 2002.

[26] D. Kuhn and M. Reilly, "An investigation of the applicability of design of experiments to software testing," in *Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop*, pp. 91–95, Los Alamitos, Calif, USA, 2003.

[27] M. B. Cohen, C. J. Colbourn, and A. C. Ling, "Constructing strength three covering arrays with augmented annealing," *Discrete Mathematics*, vol. 308, no. 13, pp. 2709–2722, 2008.

[28] A. S. Hedayat, N. J. A. Sloane, and J. Stufken, "Orthogonal arrays: theory and applications," *Technometrics*, vol. 42, no. 4, pp. 440-440, 2000.

[29] K. A. Bush, "Orthogonal arrays of index unity," *Annals of Mathematical Statistics*, vol. 23, pp. 426–434, 1952.