Research Journal of Applied Sciences, Engineering and Technology 6(2): 309-315, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: October 31, 2012 Accepted: December 21, 2012 Published: June 10, 2013

Construction Optimal Combination Test Suite Based on Ethnic Group Evolution Algorithm

Hao Chen, Shu-Yan Wang and Xiao-Ying Pan
School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Abstract: The optimal test case suite constructing problem is defined thus: given a set of test requirements and a test suite that satisfies all test requirements, find a subset of the test suite containing a minimum number of test cases that still satisfies all test requirements. Existing methods for solving test case suite generation problem do not guarantee that obtained test suite is optimal. In this study, we propose a global optimization and generation method to construct optimal combinatorial testing data. Firstly, an encoding mechanism is used to map the combinatorial testing problem domain to a binary coding space. After that, an improving ethnic group evolution algorithm is used to search the binary coding space in order to find the optimal code schema. Finally, a decoding mechanism is used to read out the composition information of combinatorial testing data from the optimal code schema and construct optimal test case suite according to it. The simulation results show this method is simple and effective and it has the characteristics of less producing test data and time consumption.

Keywords: Combinatorial testing, ethnic group evolution computing model, optimal test case suite, test data construction algorithm

INTRODUCTION

The combinatorial software testing is a method for designing test suite for the Software Under Test (SUT), which generate test case based on a certain combinatorial covering criterion. According to the difference of covering strength, the combinatorial software testing method can be classified into single factor covering method, pair wise combinatorial covering method and multiple combinatorial covering methods. All of above test methods try to make use of as little as possible test cases to cover as much as possible combinatorial sets. Since the costs of executing test cases and managing test suites may often be quite significant, a test suite subset that can still satisfy all requirements is desirable. Such a subset is known as a representative set. Assuming that the cost of executing and managing each test case is the same, a representative set with a minimum number of test cases is desirable and is called an optimal test case suite. As mentioned in Harrold *et al*. (1993), the optimal test case suite generation problem is NP-complete and as mentioned in Yan and Zhang (2009) it is equivalent to solving the set-covering problem. Cohen *et al.* (2003) gives the definition of Covering Array (CA) and Mixed level Covering Array (MCA). The difference between CA and MCA is that each factors of CA has a same value range, but in MCA it can be different. So, the CA can be looked upon as a special case of MCA and the processing method of them has no difference.

The conventional construction mechanism makes use of some mathematical methods, such as orthogonal array (Yan and Zhang, 2008) and heuristic algorithms to generate an approximate test suite. For heuristic algorithms can generate less test data than mathematical construction methods, so many researchers are absorbed in using heuristic algorithm to generate combinatorial test suite. In such studies, the one-test-at-a-time mechanism has gotten a wide application in helping a heuristic algorithm to generate test data. Based on this mechanism, in once computation, a heuristic algorithm will select a best test case t_i, which can cover most strength t combinations in the uncovering Combination Set (CS) and make it join Test Suite (TS). Then, these strength t combinations, covered by t_i, will be deleted from CS. After that, this process will repeat until all of strength t combinations are covered. There are two studies (Shiba *et al*., 2004; Zha *et al*., 2010) provide heuristic methods based on one-test-at-a-time mechanism to construct combinatorial test data. The main steps of this mechanism are as follow:

Algorithm 1: The one-test-at-a-time mechanism:

01: Initializing test suite TS = Ø
02: Initializing combination set CS according to CA

Corresponding Author: Hao Chen, School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China

Table 1: A system with four parameters

	Parameter			
Value	A	B	C	D
0	a0	b0	c0	d0
1	a1	b1	c1	d1
2	a2	b2	c2	d2

Table 2: An optimal test suite covering all pairs

ID	A	B	C	D
1	a0	b0	c0	d0
2	a0	b2	c2	d1
3	a0	b1	c1	d2
4	a1	b0	c2	d2
5	a1	b1	c0	d1
6	a1	b2	c1	d0
7	a2	b0	c1	d1
8	a2	b2	c0	d2
9	a2	b1	c2	d0

03: While (CS ≠ Ø)
04: A test case t_i has been generated by heuristic algorithm and takes it into TS
05: Updating CS
06: End While

For the one-test-at-a-time mechanism just can generate one test case in once computation process and it costs a lot of calculation on matrix transformation operation. So, it is likely to take a long time to generate whole representative set. Moreover, even to a small-scale CA, because of the one-test-at-a-time mechanism does not have the ability of dynamically adjusting the composition of TS in whole construction process. Therefore, these heuristic methods can only generate an approximate representative set generally. So, heuristic methods are difficult to generate an optimal combinatorial test suite. For example, there is a CA with four factors in Table 1 and 2 is an optimal pair wise combinatorial covering test suite of it. If we use one-test-at-a-time method to generate the combinatorial test data, it is likely to select a0b0c0d0, a1b1c1d1 and a2b2c2d2 to join TS one by one. Because of both of them can cover 6 strength 2 combinations, which suffice for the optimality criterion. On this condition, no matter whatever test case in left 78 cases has been chosen to join TS in next computation cycle, there is at least one combination is repetitive with the covered 24 strength 2 combinations generated by above three test cases. The scale of final combinatorial test suite will be more than 9. And it is no way to generate an optimal combinatorial test suite.

THE GLOBAL OPTIMIZATION AND GENERATION MECHANISM FOR TEST DATA

In this study, we will translate this problem into a code optimization problem and use the Evolution Algorithm (EA) to optimize the structure of test data. Firstly, we map the combinatorial test problem domain to a binary coding space by an encoding process. Then an EA has been used to search the coding space in order to find the optimal individual. After that, the binary code of the optimal individual has been decoded to generate combinatorial test suite by a decoding process. The core steps of this method as follow:

Algorithm 2: The EA based combinatorial test data global optimization and generation mechanism:

01: Mapping the combinatorial test problem domain to a binary code space based on encoding process
02: Initializing the population
03: While (the termination criteria aren't reached)
04: Using an EA to search the coding space
05: End While
06: Decoding the code of optimal individual in population and generating a combinatorial test suite

As we can see, encoding process, decoding process and global searching process are three main processes in this mechanism. In this section we will discuss the encoding mechanism, decoding mechanism and the fitness function.

Encoding mechanism: The aim of encode is to set the mapping relationship between the combinatorial test problem domain and the binary coding. Firstly, we give some basic definitions that are used throughout.

Definition 1: A population of EA consists of an n-tuple of strings A_i (i = 0, 1, … n-1) of length L, where the genes $\Gamma_j \in \{0, 1\}$, j = 0, 1, …L-1.

Definition 2: The problem domain of CA (N; t, k, v) is Φ and its scale is $|\Phi| = v^k$.

Definition 3: Make a serial number for all test cases in Φ by ascending order and set the value of serial number is from 0 to |Φ|-1. Then we can reference a test case in Φ by its serial number, that is $t_j \in \Phi$, j = 0, 1,…, |Φ|-1.

If we set $L = |\Phi|$, the $t_j \in \Phi$ can correspond to the gene Γ_j in the binary code by the serial number j. Moreover, we set a given, if $\Gamma_j = 1$ then the t_j will be chosen to join the test suite. According to these definitions, the gene structure of A_i can be translated into a subset of test case. For example, there are 81 test cases in the CA of Table 1. The test case of this CA are a0b0c0d0, a0b0c0d1, … and a2b2c2d2 and its serial number are 0, 1, … and 80. If the value of genes 2, 16, 41 and 77 in an individual are 1, then the corresponding test cases are t_2, t_{16}, t_{41} and t_{77}. And its details are a0b0c0d2, a0b1c2d1, a1b1c1d2 and a2b2c1d2.

Table 3: The corresponding relation between the detail of each test case and its serial number

A	B	C	D	Serial number j
0	0	0	0	0
0	0	0	1	1
0	0	0	2	2
0	0	1	0	3
…	….	…	…	…
2	2	2	2	80

Decoding mechanism: The role of decoding mechanism is to parse the coding structure and construct the combinatorial test data according to it. In order to facilitate the processing, we use a v's digit number to present the details of a test case.

Definition 4: To a covering array CA (N; t, k, v), a test case can be expressed as a *v*'s digit number and each digit correspond to a factor in the CA. The value of jth factor is k_j and the range of each factor is set from 0 to *v*-1. The function of translating *v*'s digit number of test case t_j into its serial number can be determined by:

$$j = k_j*v^j + \ldots + k_1*v^1 + k_0*v^0 \quad (1)$$

For example, to the CA in Table 1, we set the order of *v*'s digit number from top digit to low digit is corresponding to the factor from A to D. Then the corresponding relation between the details of each test case with 3's digit number and its serial number with ten's digit is shown in Table 3. The translating function is:

$$j = A_j*3^3 + B_j*3^2 + C_j*3^1 + D_j*3^0 \quad (2)$$

So, we can make use of an inverse process of formula (1) to get the details of each test case according to its serial number. The decoding process for a CA with *k* factors is:

Algorithm 3: Decoding mechanism:

01: Get the coding information of A_i
02: For $j = 0$ to L
03: If (Γ_j == 1) //Decoding and getting the details of t_j
04: dnum = j, $m = 0$
05: While (dnum>0)
06: x = dnum%v; dnum/ = v
07: $z_{j,m} = x$; //x is the value of mth factor in t_j
08: m++
09: End While
10: End If
11: End For

Fitness function: Generally, we can evaluate the price of A_i from two aspects in the iteration searching process, the first one is the covering degree for the strength t combination and the second one is the scale of combinatorial test suite. In this study, the fitness of A_i is determined by:

$$\text{Fitness} = \omega*\mu-\theta \quad (3)$$

where,
ω = The number of covered combination in CS
θ = The scale of combinatorial test suite

Obviously, $0<\omega\leq v^tC^t_k$ and $0<\theta\leq|\Phi|$, $\mu>1$ is a controlled parameter, which is set to make sure the Fitness>0.

CONSTRUCTING TEST SUITE BASED ON EGEA/PAD

From Algorithm 2, we can see the quality of combinatorial test data is based on the optimization performance of EA. In this section, we produce an improving Ethnic Group Evolution Algorithm (EGEA) to search binary code space and find high quality solution.

As we known, the population structure of EA has a heavy influence on its searching efficiency. So, we propose a novel population searching mechanism EGEA, which makes use of the clustering process to analyze the population structure and build up ordered ethnic group organization to control the population searching process (Hao *et al.*, 2010). The experiment has shown it is helpful in avoiding the premature convergence phenomenon while increasing the convergence speed of population greatly.

In ethnic group clustering process, individuals have been assigned into ethnic group so that they have a high degree of similarity within the ethnic group and that the ethnic group is to be distinct. The clustering model consists of two parts: a technique for calculating distance for binary code between two individuals and a grouping technique to minimize the distance between individuals of each ethnic group. The objective here and in any clustering method, is to minimize the distance between individuals in each ethnic group while maximizing the distance between ethnic groups.

In order to design suitable clustering method, we need to analyze the bound characteristics of set covering problem. The following inequalities on CAN (*t*, *k*, *v*) are basic ones and it can be found in Chateauneuf and Kreher (2002) and Martirosyan and Tran Van (2004):
Symbol-fusing:

$$\text{CAN}(t, k, v\text{-}1) \leq \text{CAN}(t, k, v) \quad (4)$$

Row-deleting:

$$\text{CAN}(t, k\text{-}1, v) \leq \text{CAN}(t, k, v) \quad (5)$$

The lower and upper bound: for any $v\geq 2$, $t\geq 2$ we have:

$$v^t \leq \text{CAN}(t, k, v) \leq 2^t.v^t\text{-}1 \quad (6)$$

where, $k\leq 2^n$ and n is the smallest integer such that $v\leq 2^n$.

From these inequalities we can see, the genes, whose value are 1, have occupied a small proportion in the optimal code schema. In order to emphasize importance of these genes, we produce a novel hierarchical ethnic group clustering method based On Positive Attribute Distance (PAD).

Calculating the similarity of individuals by PAD: In EGEA, we calculate the binary code distance between two individuals based on the Weighted Hamming Distance (WHD) during the ethnic group clustering process. The WHD between A_i and A_j is:

$$\delta_{ij} = \sum_{w=1}^{m}\left[\sum_{k=1}^{l_w}\left(b_{w,k} \times \eta_{w,k}\right)\right] \tag{7}$$

where,

$$b_{w,k} = \begin{cases} 1, a_{i,w,k} \neq a_{j,w,k} \\ 0, a_{i,w,k} = a_{j,w,k} \end{cases}$$

and the weight of gene is $\eta_{w\,k} = L_w - k_w + 1$. Meanwhile, the similarity index between two individuals is as follows:

$$D(A_i, A_j) = D_{ij} = \delta_{ij} \Big/ \sum_{w=1}^{m}\left(\sum_{k=1}^{L_w}\eta_{w,k}\right) \tag{8}$$

The result obtained is thus an expression between 0 and 1, where 1 designates absolute similarity between the two binary codes of individuals and 0 designates absolute diversity between the two binary codes of individuals.

In study (Gelbard and Spiegler, 2000; Gelbard *et al*., 2007), propose to use PAD to calculate the distance between objects. As in the HD, the PAD yields a clustering method by calculating the degree of similarity between objects whose various features are represented in a binary manner. Unlike HD, however, PAD follows the outcome of the Hempel's Raven paradox regarding the use of positive predicates. In PAD, the similarity between two binary sequences is as follows:

$$0 \leq \frac{2\psi_{ij}}{\psi_i + \psi_j} \leq 1 \quad , \quad \psi_i + \psi_j > 0 \tag{9}$$

where,

ψ_i = The number of 1's in i^{th} binary sequences
ψ_j = The number of 1's in j^{th} binary sequences
ψ_{ij} = The number of 1's common to both i^{th} and j^{th} Binary sequences

Since the definition of PAD is more simple than WHD. Meanwhile, it is easy to calculate and the result of PAD is in the interval between 0 and 1, where 1 expresses absolute similarity and 0 expresses absolute diversity. These properties make PAD can take place of WHD easily. So, we try to make use of PAD as the similarity index between individuals for ethnic group clustering.

Ethnic group hierarchical clustering based on PAD: In this study, we propose a new ethnic group hierarchical clustering mechanism based on PAD and the details are shown in Algorithm 4:

Algorithm 4: Ethnic Group Hierarchical Clustering:

Step 1: **Initialization:** Selecting a part of representative individuals, named as macrogamete M_i (i = 1, 2, …, |M|), from population to create |M| independent groups and set each M_i to be the center individual of each group O_i (i = 1, 2, …, |M|).

Step 2: Calculating PAD among macrogamete and saving these information in a table T.

Step 3: Finding two groups O_r and O_k, which have the minimal PAD between their center individuals from T:

$$(O_r, O_k) = \min\{PAD(O_i, O_j)\}, \quad i \neq j \quad and \quad i, j = 1,2,...,|O| \tag{10}$$

Step 4: **Consolidation:** If PAD (O_r, O_k) <θ, then consolidate O_r and O_k into a new group O_{rk} and set the individual with the maximal race exponent to be the new center individual of O_{rk}.

Step 5: If there is a new group who has been generated in step 4 then return to step 3.

Step 6: **Creating ethnic group:** Transform each O to be a ethnic group $O_i \rightarrow E^i$ and set the center individual of O_i to be the center individual of E^i.

Step 7: **Setting ethnic group weight:** Sort E^i according to the race exponent of its center individual. If the sequence number of E^i in this queue is j, then the weight of E^i is:

$$h^i = 2(|E| - j + 1) \Big/ \left(|E|^2 + |E|\right), \quad i, j = 1,2,...,|E| \tag{11}$$

The parameter θ∈ (0, 1) has been set up for controlling the clustering granularity. For it affect the structure of ethnic group directly, we hope θ can dynamically adjust its value according to the status of population. So, a adaptive mechanism has been designed to adjust the number of θ. The formula is:

$$\theta = (\gamma + \varepsilon)/2 \tag{12}$$

where,

γ = Diversity parameter of population and ε∈ (0, 1) is a constant

SIMULATION RESULTS

In this section, there are four serials and 19 CA problems are selected for simulation tests. Based on Algorithm 2, we make use of EGEA/PAD and CGA to search the binary code space. Both the source codes are realized by VC++6.0 on a 2.1-GHz AMD Phenom PC with 1 GB memory and the operation system are Windows 2003.

The experimental statistic results of EGEA/PAD and CGA are obtained over 20 independent trials. In first round experiments, we set t = 2, v = 2 and k is gradually increasing from 3 to 9 and the statistic results are shown in Table 4. In second round experiments, we set t = 2, v = 3 and the k is gradually increasing from 3 to 7 and the statistic results are shown in Table 5. In third round experiments, we set t = 3, v = 2 and the k is gradually increasing from 4 to 8 and the statistic results are shown in Table 6. In finally round experiments, we set t = 3, v = 3 and the k is gradually increasing from 4 to 7 and the statistic results are shown in Table 7. So far the best result of these problems can be found in Colbourn's web site (Colbourn, 2009).

As we can see, the EGEA/PAD and CGA can find the best result when v = 2, t = 2 and k is less than 9 and 7, meanwhile, the EGEA/PAD and CGA can find the

Table 4: The statistic results of EGEA/PAD and CGA on the scale of test data when t = 2 and v = 2

				The scale of test data generated by CGA				The scale of test data by EGEA/PAD				
k	Φ	Combination scale	Best result	Max.	Min.	Avg.	St.	Max.	Min.	Avg.	St.	Generation
3	8	12	4	4	4	4	0	4	4	4	0	20
4	16	24	5	5	5	5	0	5	5	5	0	20
5	32	40	6	6	6	6	0	6	6	6	0	50
6	64	60	6	6	6	6	0	6	6	6	0	200
7	128	84	6	12	6	9.4	2.48	6	6	6	0	500
8	256	112	6	18	6	11.3	4.37	12	6	9.5	2.67	1000
9	512	144	6	66	42	52.9	11.80	41	28	31.2	8.32	2000

Max.: Maximum; Min.: Minimum; Avg.: Average

Table 5: The statistic results of EGEA/PAD and CGA on the scale of test data when t = 2 and v = 3

				The scale of test data generated by CGA				The scale of test data by EGEA/PAD				
k	Φ	Combination scale	Best result	Max.	Min.	Avg.	St.	Max.	Min.	Avg.	St.	Generation
3	27	27	9	9	9	9	0	9	9	9	0	100
4	81	54	9	9	9	9	0	9	9	9	0	500
5	243	90	11	18	11	16.7	2.64	13	11	11.4	0.33	1000
6	729	135	12	20	15	18.3	2.25	19	13	15.7	2.67	2000
7	2187	189	12	265	207	235.5	20.80	194	153	179.2	14.85	2000

Max.: Maximum; Min.: Minimum; Avg.: Average

Table 6: The statistic results of EGEA/PAD and CGA on the scale of test data when *t* = 3 and *v* = 2

				The scale of test data generated by CGA				The scale of test data by EGEA/PAD				
k	Φ	Combination scale	Best result	Max.	Min.	Avg.	St.	Max.	Min.	Avg.	St.	Generation
4	16	32	8	8	8	8	0	8	8	8	0	100
5	32	80	10	13	10	11.2	1.16	11	10	9.3	0.58	200
6	64	160	12	17	15	16.4	1.24	14	12	12.5	0.69	500
7	128	280	12	22	17	19.4	2.31	17	12	13.2	2.09	1000
8	256	448	12	63	42	54.2	14.60	41	38	37.2	1.38	2000

Max.: Maximum; Min.: Minimum; Avg.: Average

Table 7: The statistic results of EGEA/PAD and CGA on the scale of test data when *t* = 3 and *v* = 3

				The scale of test data generated by CGA				The scale of test data by EGEA/PAD				
k	Φ	Combination scale	Best result	Max.	Min.	Avg.	St.	Max.	Min.	Avg.	St.	Generation
4	81	108	27	31	29	30	0.89	29	27	27.5	0.67	500
5	243	270	33	51	39	44	4.31	45	33	36.2	4.82	1000
6	729	540	33	108	96	101.5	5.89	59	42	47.9	5.16	2000
7	2187	945	40	356	296	335.6	28.90	254	193	215.4	20.68	2000

Max.: Maximum; Min.: Minimum; Avg.: Average

Table 8: Comparison between EGEA/PAD and another two methods on the minimum scale of test data when *t* = 2

v		2							3				
k		3	4	5	6	7	8	9	3	4	5	6	7
Hartman *et al.* (2004)	The min scale of test data	4	5	6	6	6	6	6	9	9	15	15	15
Williams *et al.* (2002)	The min scale of test data	4	5	6	6	6	-	-	-	9	13	-	-
	Time (s)	<0.01	0.01	0.70	16.57	441.21	-	-	-	0.08	-	-	-
EGEA/PAD	The min scale of test data	4	5	6	6	6	6	28	9	9	11	13	153
	Time (s)	<0.01	<0.01	0.04	4.09	54.56	122.4	251.2	0.05	0.10	68.9	205.6	554.5

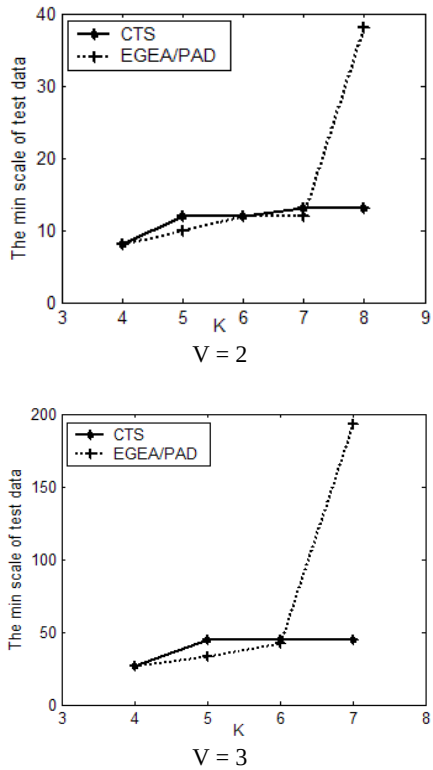

Fig. 1: Comparison between EGEA/PAD and CTS on the minimum scale of test data with $t = 3$

best result when v up to 3 and k is less than 7 and 6. If t is up to 3, the EGEA/PAD and CGA can find the best result when v = 2 and k is less than 8 and 6, but only the EGEA/PAD can find the best result when v = 3 and k is less than 6. The result show an EA based combinatorial test data global optimization and generation mechanism is valid. It can find most optimal results in 19 CA problems. Meanwhile, the performance of searching algorithm have a heavily influence on the quality of solution. The comparison of statistic results between EGEA/PAD and CGA show the EGEA/PAD can improve the quality of solution greatly.

In study (Alan and Leonid, 2004), Hartman use the Combinatorial Test Services (CTS) package to solve CA problems. Williams translate optimal combinatorial test suite construction problem into an integer program problem and list the minimum scale of test data in study (Williams and Probert, 2002). The comparison is shown in Table 8. And the run times are also compared.

Meanwhile, the comparison between EGEA/PAD and CTS on the test data minimum scale when t = 3 is shown in Fig. 1. As can be seen, the EGEA/PAD significantly outperforms another two methods for 10 CA problems in all 12 problems when t = 2, meanwhile, it also has get the best results for 7 CA problems in all 9 problems when t = 3.

CONCLUSION

In this study, we propose a combinatorial test data global optimization and generation method, which include the encoding and decoding mechanism and an improving ethnic group evolution algorithm-EGEA/PAD. The experimental results show this mechanism have a good performance in most testing problem.

However, the problem scale of CA is growing exponentially and it restrains the searching ability of this method heavily. In future study, we will focus on design more succinct coding mechanism or coding compression mechanism to make this method can solve more large-scale and complex CA problem.

ACKNOWLEDGMENT

This study is supported by the National Natural Science Foundation of China under Grant No. 61050003.

REFERENCES

Alan, H. and R. Leonid, 2004. Problems and algorithms for covering arrays. Discrete Math., 284: 149-156.

Chateauneuf, M. and D.L. Kreher, 2002. On the state of Strength-Three Covering Arrays. J. Combin. Des., 10: 217-238.

Cohen, M.B., P.B. Gibbons, W.B. Mugridge and C.J. Colbourn, 2003. Constructing test suites for interaction testing. Proceeding of the International Conference on Software Engineering, pp: 38-48.

Colbourn, C.J., 2009. CA Tables for t = 2, 3, 4, 5, 6. Retrieved from: http://www.public.asu. edu/~ ccolbou/src/tabby/catable.html.

Gelbard, R. and I. Spiegler, 2000. Hempel's raven paradox: A positive approach to cluster analysis. Comput. Oper. Res., 27(4): 305-320.

Gelbard, R., O. Goldman and I. Spiegler, 2007. Investigating diversity of clustering methods: An empirical comparison. Data Knowl. Eng., 63(1): 155-166.

Hao, C., C. Du-Wu, C. Yin-An, T. Yong-Qin and L. Kun, 2010. Ethnic group evolution algorithm. Chin. J. Softw., 21(5): 978-990.

Harrold, M.J., R. Gupta and M.L. Soffa, 1993. A methodology for controlling the size of a test suite. ACM T. Softw. Eng. Methodol., 2(3): 270-285.

Martirosyan, S. and T. Tran Van, 2004. On t-covering array. Designs Code Cryptogr., 32: 323-339.

Shiba, T., T. Tsuchiya and T. Kikuno, 2004. Using artificial life techniques to generate test cases for combinatorial testing. Proceeding of the IEEE Annual International Conference on Computer Software and Applications, pp: 72-77.

Williams, A.W. and R.L. Probert, 2002. Formulation of the interaction test coverage problem as an integer program. Proceeding of the 14th International Conference on the Testing of Communicating Systems, pp: 283-298.

Yan, J. and J. Zhang, 2008. A backtracking search tool for constructing combinatorial test suites. J. Syst. Softw., 81(10): 1681-1693.

Yan, J. and J. Zhang, 2009. Combinatorial testing: Principles and methods. Chin. J. Softw., 20(6): 1393-1405.

Zha, R.J., D.P. Zhang, C.H. Nie and B.W. Xu, 2010. Test data generation algorithms of combinatorial testing and comparison based on cross-entropy and particle swarm optimization method. Chin. J. Comput., 33(10): 1896-1908.