Open Access **Baghdad Science Journal** **P-ISSN: 2078-8665**
2020, 17(3) Supplement (September):1002-1009 **E-ISSN: 2411-7986**

DOI: http://dx.doi.org/10.21123/bsj.2020.17.3(Suppl.).1002

CTJ: Input-Output Based Relation Combinatorial Testing Strategy Using Jaya Algorithm

Mohammed Issam Younis [*1] ***Abdul Rahman A. Alsewari*** [2] ***Ng Yeong Khang*** [2]
Kamal Z. Zamli [2]

[1]Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq;
younismi@coeng.uobaghdad.edu.iq
[2]Faculty of Computing, Universiti Malaysia Pahang, Pahang, Malaysia;
*Corresponding author: younismi@gmail.com; younismi@coeng.uobaghdad.edu.iq, Tel.: +9647714195762
[*]ORCID ID: 0000-0003-4884-3747
Other e-mails: alsewari@ump.edu.my, ngyk95@gmail.com, kamalz@ump.edu.my
Other ORCID IDs: 0000-0002-7802-6628, 0000-0001-5054-2594,0000-0003-4626-0513

Received 15/7/2019, Accepted 10/3/2020, Published 8/9/2020

Abstract:

Software testing is a vital part of the software development life cycle. In many cases, the system under test has more than one input making the testing efforts for every exhaustive combination impossible (i.e. the time of execution of the test case can be outrageously long). Combinatorial testing offers an alternative to exhaustive testing via considering the interaction of input values for every t-way combination between parameters. Combinatorial testing can be divided into three types which are uniform strength interaction, variable strength interaction and input-output based relation (IOR). IOR combinatorial testing only tests for the important combinations selected by the tester. Most of the researches in combinatorial testing applied the uniform and the variable interaction strength, however, there is still a lack of work addressing IOR. In this paper, a Jaya algorithm is proposed as an optimization algorithm engine to construct a test list based on IOR in the proposed combinatorial test list generator strategy into a tool called CTJ. The result of applying the Jaya algorithm in input-output based combinatorial testing is acceptable since it produces a nearly optimum number of test cases in a satisfactory time range.

Key words: Jaya algorithm, software testing, combinatorial testing, t-way testing, system reliability

Introduction:

Combinatorial testing is a black-box testing technique that generates test cases by combining the values of different input parameters using combinatorial optimization strategies to reduce the interfaces fail and increase the reliability of the system (1). Taking the study from the failure of the medical device application, the failure-triggering fault interaction (FTFI) is 68% for the single parameter value, 97% of failures triggered by 2 combination values while the percentage of failures caused by 3 and 4 combination values are 99% and 100% respectively (2). By using combinatorial testing, all input values of the test objects and interactions between each parameter are tested. This testing causes a higher detection of interaction failures compared to single parameter testing.

More specifically, combinatorial test generation relates to the process of searching the optimum number of test cases for test consideration based on the interaction of t-way parameters (where t indicates the interaction strength). Many different optimization strategies are used to generate the test cases for combinatorial testing such as Harmony Search (3), Genetic Algorithm (4), Ant Colony Algorithm (4), Simplified Swarm Optimization (5), Differential Evolution Algorithm (6) and so on.

Most of these aforementioned algorithms have control parameters. For example, Harmony Search requires 4 parameter controls namely maximum

iteration, harmony size, harmony memory consideration rate, and pitch adjustment. Similarly, Ant Colony adopts maximum iteration, population size, evaporation rate, pheromone influence, and heuristic influence as the parameter controls. Tuning of these control parameters is necessary to ensure the balance between exploration (i.e. sufficiently roaming through all the potential areas in the search space) and exploitation (i.e. searching around the known best). Often, the tuning of these control parameters is difficult as poor tuning can lead to falling into local optima and unnecessary increase in computational performance.

Addressing these issues, our work proposes to adopt the Jaya algorithm as the backbone algorithm for a combinatorial optimization problem. Apart from being a significantly new algorithm, the Jaya algorithm (7) has only two parameter controls (i.e. common controls to all search algorithms: population size and maximum iteration). This is an important feature as the Jaya algorithm offers a fair learning curve as well as straightforward adoption effort as there is no need for significant tuning (8, 9). Additionally, the ease of resolving discrete optimization problems and convergence to global optimum value make the Jaya algorithm even better than other optimization algorithms (10). Furthermore, the Jaya algorithm offers a faster convergence speed than that of the TLBO algorithm. Hence, a new strategy in overcoming input-output based relation combinatorial testing using Jaya algorithm called CTJ is introduced. Summing up, the main contribution of the work can be summarized as follows:

- Unlike the state-of-the-art implementation, CTJ is the first implementation that adopts parameter-free algorithm as the backbone algorithm.
- CTJ also supports combinatorial input-output based relation.
- CTJ performs well when compared with the state-of-the-art implementations.

Related Work:

This section contains the mathematical notation of Input-output based relation combinatorial testing, followed by a survey on combinatorial testing.

To express IOR in a mathematical form, a combination of input-output relationship (Rel) and covering array is needed to come out with input-output based relations covering array. Rel can be written in this form, Rel = {{x1}, {x2}, … ,{xn}} where x is the combination of inputs that will generate the specific output. Input-output based relations covering array, IOR (N, C, Rel) is the mathematical form of IOR relation. N represents the number of test case in the test suite, C is the number of value of each input parameter (v1P1, v2P2, … , vnPn) where v is the amount of input value and p is the amount of parameter that has the same amount of v while Rel is the input-output relationship as stated above (11).

The greedy algorithm has previously been used by Schroeder, the researcher who proposed combinatorial testing with IOR feature in one of his studies to determine particular combinations of inputs which affect the outputs of the program (12). Furthermore, the study of the variable strength combinatorial test suite using the Greedy algorithm is done by Wang and his colleagues to increase the flexibility of controlling the interaction strength (13).

The density-based algorithm was first applied in the research in interaction testing by Colbourn and his associates, but it was limited only to pairwise testing (14). Colbourn and his colleagues then continued the research by increasing the limit of interaction strength (15). There is another research conducted by Wang and his colleagues that introduced variable strength interaction in combinatorial testing using density as the optimization algorithm (13).

AURA is a non-deterministic input-output based relationship combinatorial testing strategy proposed by Ong and Kamal (16). This strategy is focusing on solving the mapping of symbolic values to actual data manually and the lack of flexibility of existing test suite generation. AURA supports uniform interaction strength and variable interaction strength with the strength up to 3 as well as input-output based relationships.

There are several types of research have been done on showing the implementation of the Ant Colony Optimization algorithm in combinatorial testing. These include the study of comparing the efficiency of ACO with simulated annealing and genetic algorithm (17) as well as applying ACO in variable interaction strength combinatorial testing (18). ACO has been proposed to be applied in IOR combinatorial testing in recent research by Ramli and her associates (19) but there is a lack of results of implementation in the study.

Genetic Algorithm has been implemented in solving combinatorial testing optimization problems such as in the research by Shiba and his associates (4). Srivastava and Kim developed variable strength interaction combinatorial testing using GA to focus on the parts that are critical by implementing a more selective approach (20). Furthermore, McCaffrey conducted a study to identify the effectiveness of GA in pairwise testing

(21). Nevertheless, GA has not yet been applied for the optimization in combinatorial testing that has an IOR feature.

Several types of research have applied the Harmony Search algorithm in combinatorial testing. In 2011, Alsewari applied the HS algorithm in t-way interaction test data generation (22). Besides, the HS algorithm is being implemented in pairwise testing strategy (PHSS) and PHSS is outperformed existing strategies in terms of the size of the test suite generated in the study in (23). Besides, utilization of HS algorithm in variable strength interaction combinatorial testing with constraint support is carried out by (3). However, to the best of the authors' knowledge, there is no research deals with the IOR feature using HS.

Based on the findings, Particle Swarm Optimization has been implemented in pairwise testing by Chinese researchers in 2010 (24). Other than that, test suite generation using variable interaction strength also implemented PSO to solve the combinatorial problem (25). Still, there is no combinatorial testing that employs the IOR feature using PSO.

There are few types of research related to combinatorial testing using Simulated Annealing for optimization that have been carried out. Cohen and Colbourn used SA to solve the optimization problem while constructing a test suite for interaction testing (26). SA also is used to combine with algebraic construction to build covering arrays for interaction testing that is strength three (27). Again, no research uses the SA algorithm to optimize the combinatorial problem in combinatorial testing that features IOR.

CTJ Implementation:

This section gives a details description of the proposed Jaya algorithm. To realize the input-output based relation combinatorial testing based on Jaya algorithm, there are five levels of actions to be carried out. The five-level actions start by reading the input values entered by the user, data analyzation and data mapping, input values combination generation, test case generation, and final test suite generation.

Step1: Data Analysis and Data Mapping

The purpose of the analysis is to ensure the information user keyed in is in the right format and syntax. Any wrong information inclusive of amiss format and syntax entered will cause the system not able to recognize even more the system will crash. Therefore, preventive action is taken to counter the happening of the above situation by giving users feedback messages so that they can recheck the problem and make the correction.

Data mapping is carried out right after data analysis. The input values from each parameter that has been verified during the data analysis process will undergo the mapping process with integers. By applying data mapping, the time taken to generate all possible combinations of input values and the test case will be reduced due to the size of the input data is decreased. Often, the size of a string is larger than an integer. Smaller byte of data always processes faster than the larger one. Hence, the string values of input data are being substituted with integers during data processing.

Step2: Combinations of Input Value Generation

In this step, the combination of input values is generated to be used in test case generation based on the interaction strength or based on the selection input/output combinations. Each input value that belongs to the same parameter and has mapped to the corresponding integer is merged with other values of parameters to form combinations of values. There are two types of combinations implemented which are combinations of input values based on input-output relationship and interaction strength.

Step3: Test Case Generation Based on Jaya Algorithm

After gathering all combinations of input values, the next step is to generate the test case. It starts with generating a test case by randomly pick one of the input values from each parameter. The generated test case which will be assessed by determining the number of combinations of input values that generated in Step2 covered by the test case. The best and worst test cases in terms of coverage in the population will be picked for modification purposes. Each test case in the population will be improved by applying the modification formula in the Jaya algorithm (Eq. (1)) based on the best and worst test cases. If the test case generated after employing the modification has better coverage than the previous one, it will then replace the former test case. After one iteration, the best and worst test cases will be re-selected, and the modification is done based on the new best and worst test cases.

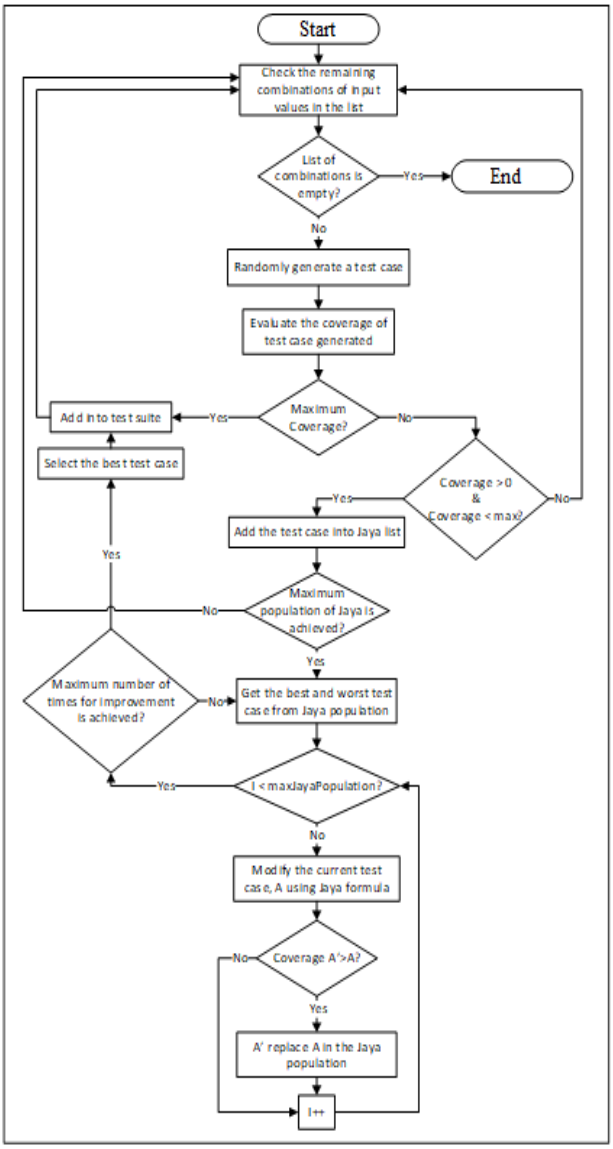

Figure 1. The flowchart of test case generation using Jaya algorithm.

The process is iterated until the maximum number of generations is achieved. The best test case generated at the end of the iterative process is added into a temporary test suite. The whole process keeps repeating until all combinations of input values are fully covered as illustrated in Fig. 1.

Assume that the best candidate solution (best) has the best value of f(x) while the worst candidate solution (worst) has the worst value of f(x) in the candidate suite, and Xj,k,i is the value under the j^{th} design variable of the kth candidate solution for the i^{th} iteration. Thus, Eq. (1) is fulfilled the requirements.

$$X'_{j,k,i} = X_{j,k,i} + r_{1,k,i}\left(X_{j,best,i} - |X_{j,worst,i}|\right) - r_{2,j,i}\left(|X_{j,worst,i}| - X_{j,k,i}\right) \quad \textbf{Equation (1)}$$

Experimental Results and Discussion:

This section is divided into three parts which are parameter tuning of CTJ, two experiments for IOR and one experiment for uniform strength interaction strength to evaluate the difference between CTJ with another existing strategy in handling combinatorial optimization.

Parameter Tuning of CTJ

All experiments conducted are using Intel i7-6500U as the CPU with the RAM of 8GB in Windows 10 Professional operating system. There are only two common controlling parameters involved in CTJ which are population size and number of iterations. Hence, the tuning of parameters setting is performed to ensure the optimal results and efficiency of CTJ before the experiments are carried out. The tuning process is executed using one of the configurations from each IOR and uniform interaction strength experiments in (28) and (22). Three types of parameter settings stated in Table 1 are being experimented for 10 iterations to figure out which parameter setting will generate the most optimum and efficient result.

Table 1. Parameters settings.

Parameters Setting	Population Size	Number of Iterations
S1	10	100
S2	50	500
S3	100	1000

The first experiment is adapted from (28) by using 10 parameters with 3 values each and the first 30th input-output relationships are utilized. From the result of the execution in Table 2, S1 setting is generated test cases in the shortest time but it yielded the highest number of test cases generated. S3 has the best number of generated test cases but it consumed a very long time to finish the execution. If compared to S2, S3 took approximately four times of S2 time to reduce five test cases to be generated in the best result. It is impractical to consume such a long time to reduce a small number of test cases. The number of test cases produced in S1 is reduced significantly compared to S2 which reduced 18 test cases. This result is much more optimum and acceptable to be used.

Table 2. Test case size and execution time in 30 input-output relationships configuration.

Parameters Setting	Number of Test Cases		Average Execution Time (seconds)
	Best	Average	
S1	136	139.6	63.089
S2	118	122.5	1148.0973
S3	113	116.6	4337.3287

The second configuration of experiment is originated from (22) and the configuration is 6 parameters with 6 values each with uniform interaction strength of 3. The best and average number of test cases generated as well as the average execution time are stated in Table 3. S1 in this experiment is still the fastest parameter setting that completed the test case generation. However, the number of test cases it produced is still undesired compared to S2 and S3 settings. The time taken for the S3 setting to complete the generation of test cases is approximately 3 hours while S2 only took 55 minutes. The difference in the number of test cases generated between S3 and S2 is just eight test cases. These issues show that it is unrealistic to use an S3 setting in the real environment.

Based on both experiments that are conducted to decide the parameter settings, the S2 setting is selected to be parameter setting for all experiments since it is capable to generate the optimum number of test cases in a satisfactory time frame.

Table 3. Test case size and execution time in CA (N; 3, 6^6) configuration.

Parameters Setting	Number of Test Cases		Average Execution Time (seconds)
	Best	Average	
S1	387	396.9	215.0552
S2	354	358.8	3270.3555
S3	346	349.3	10784.2423

Experiments for Input-Output Based Relation

Two experiments published in (28) are conducted to evaluate the performance of CTJ in IOR. The strategies that have implemented the IOR feature which included in both experiments are Density (13), TVG (29), ReqOrder (30), ParaOrder (13), Union (31), Greedy (12), ITTDG (11) and AURA (16). Both experiments share the same input-output relationships, but their configuration of parameters and their values are different. The first experiment used IOR (N, 3^{10}, R) configuration while the second experiment employed IOR (N, 2^3, 3^3, 4^3, 5^1, R) configuration. All IOR relationships are stated in Table 4. The experiment will be carried out for 6 IOR configurations. Initially, the first 10 relationships will be tested as the first IOR configuration and the second 10 relationships will be added into the first IOR configuration to become the second IOR configuration for the second evaluation. For the subsequence evaluations, another 10 relationships will be added into the previous configuration until all 60 relationships are assessed.

Table 4. 60 Input-output relationships (R) that utilized in IOR experiments.

	10^{th} relationship	20^{th} relationship	30^{th} relationship	40^{th} relationship	50^{th} relationship	60^{th} relationship
Relationship (R)	{1, 2, 7, 8}	{2, 3, 4, 8}	{1,3,6,9}	{0,2,7,9}	{2,3,9}	{0,6,7,9}
	{0, 1, 2, 9}	{2, 3, 5}	{2,4,7,8}	{1,2,3}	{1,5,8}	{2,6,7,9}
	{4, 5, 7, 8}	{5, 6}	{0,2,6,9}	{1,2,6}	{1,3,5,7}	{2,6,8}
	{0, 1, 3, 9}	{0, 6, 8}	{0,1,7,8}	{2,5,9}	{0,1,2,7}	{2,3,6}
	{0, 3, 8}	{8, 9}	{0,3,7,9}	{3,6,7}	{2,4,5,7}	{1,3,7,9}
	{6, 7, 8}	{0, 5}	{3,4,7,8}	{1,2,4,7}	{1,4,5}	{2,3,7}
	{4, 9}	{1, 3, 5, 9}	{1,5,7,9}	{2,5,8}	{0,1,7,9}	{0,2,7,8}
	{1, 3, 4}	{1, 6, 7, 9}	{1,3,6,8}	{0,1,6,7}	{0,1,3,6}	{0,1,6,9}
	{0, 2, 6, 7}	{0, 4}	{1,2,5}	{3,5,8}	{1,4,8}	{1,3,7,8}
	{4, 6}	{0, 2, 3}	{3,4,5,7}	{0,1,2,8}	{3,5,7,9}	{0,1,3,7}

The result of the first experiment which used IOR (N, 3^{10}, R) configuration with the relationships in Table 4 is shown in Table 5 and the highlighted number of test cases represents the most minimum number of test cases produced out of all strategies. Overall, ITTDG is still outperformed other strategies in terms of the size of test cases generated. However, CTJ is still delivered an almost optimum solution if compared to the best result generated by ITTDG and ParaOrder. The average difference between the best result of CTJ and other strategies is five test cases only. Besides, the time of execution of CTJ is considerably fast. For R10, CTJ took only approximately 5 minutes to finish the test case generation. 7 minutes, 19 minutes, 27 minutes, 33 minutes and 35 minutes

are taken by CTJ to complete the execution of R20, R30, R40, R50 and R60 respectively.

Table 5. Test cases size and execution time of IOR (N, 3^{10}, R) configuration.

R	Density	TVG	ReqOrder	ParaOrder	Union	Greedy	ITTDG	AURA	CTJ		
									Best	Average	Average Execution Time (seconds)
10	86	86	153	105	503	104	**81**	89	88	90.3	334.604
20	95	105	148	103	858	110	**94**	99	100	101.3	444.927
30	116	125	151	117	1599	122	**114**	132	118	122.5	1148.097
40	126	135	160	**120**	2057	134	122	139	128	130.1	1660.429
50	135	139	169	148	2635	138	**131**	147	134	137.8	2006.527
60	144	150	176	142	3257	143	**141**	158	145	148.9	2128.845

Table 6 is the result of the execution of experiment two in IOR. The most minimum number of test cases produced by CTJ in R10 and R20 only vary for 5 test cases if compared to the best algorithms. While R30 to R60, the difference between the best result of CTJ and Density is not more than 12 test cases. Additionally, the time taken to complete the execution in experiment two is in the range of 500 to 1000 seconds which approximately 8 to 16 minutes only. These show CTJ generates solutions that are close to optimum.

Table 6. Test cases size and execution time of IOR (N, 2^3, 3^3, 4^3, 5^1, R) configuration.

R	Density	TVG	ReqOrder	ParaOrder	Union	Greedy	ITTDG	AURA	CTJ		
									Best	Average	Average Execution Time (seconds)
10	144	144	154	144	505	**137**	144	144	144	144.5	509.888
20	160	161	187	161	929	**158**	160	182	165	167.1	712.472
30	**165**	179	207	179	1861	181	169	200	170	173.2	699.674
40	**165**	181	203	183	2244	183	173	207	173	176	748.750
50	**182**	194	251	200	2820	198	183	222	191	194.7	842.738
60	**197**	209	250	204	3587	207	199	230	209	211.5	987.918

The number of test cases generated through CTJ is still acceptable in overall if compared to the size of the test suite produced through exhaustive testing. Furthermore, the results produced by other strategies often go for a very high population size and a more iteration for improvement of the solutions while the parameters setting of CTJ for these experiments are only 500 iterations with the population size of 50. Different parameters setting will affect how well the strategy performed and hence resulting in a different size of test suite produced.

Conclusion:

In this paper, the existing strategies that are used in solving combinatorial testing such as Greedy, Density, AURA, ACO, GA, HS, PSO and SA have been studied. CTJ which stands for the combinatorial testing strategy that featured input-output based relation using Jaya algorithm is proposed and introduced. Referring to the results of the experiments, CTJ performs with acceptable performance especially in test case generation through input-output based relation. Overall, comparing the CTJ results to the results of existing test case generation strategies, CTJ generates more test cases compared to the other strategies, but this study focuses on generating test cases with acceptable test list size within short processing time. One of the constraints or challenges that face the testers is the test list generation time. The test list generation processing time needs to be as fast as possible to conduct the testing with a short time. So, in this research the test list generation processing time is the major factor. The search engine in CTJ is the main part which is based on the Jaya algorithm. Jaya algorithm depends on the global search approach to find the best solutions which may lead to moving far from the available solutions in the population space. Based on the low

performance of the CTJ in the test case generation, the Jaya algorithm in CTJ must go through some modifications in order to improve the performance of CTJ and even generate a more optimum number of test cases. In addition, CTJ can be enhanced by adding the support of variable interaction strength, constraints, and seeding.

Acknowledgment:

This research is funded by, UMP (RDU190334), A Novel Hybrid Harmony Search Algorithm with Nomadic People Optimizer Algorithm for Global Optimization and Feature Selection, and (FRGS/1/2018/ICT05/UMP/02/1) (RDU190102), A Novel Hybrid Kidney-Inspired Algorithm for Global Optimization Enhance Kidney Algorithm for IoT Combinatorial Testing Problem.

Conflicts of Interest:

The authors declare that they have no conflicts of interest.

Authors' declaration:

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are mine ours. Besides, the Figures and images, which are not mine ours, have been given the permission for re-publication attached with the manuscript.
- The author has signed an animal welfare statement.
- Ethical Clearance: The project was approved by the local ethical committee in University of Baghdad.

References:

1. Younis MI. MVSCA: multi-valued sequence covering array. J Eng. 2019; 25 (11):82-91.DOI: 10.31026/j.eng.2019.11.0 7.
2. Kuhn DR, Wallace DR, Gallo AM. Software fault interactions and implications for software testing. IEEE T Softw Eng. 2004; 30 (6): 418-421.
3. Alsewari ARA, Zamli KZ. Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. Inform Software Tech. 2012; 54(6):553–568.
4. Shiba T, Tsuchiya T, Kikuno T. Using artificial life techniques to generate test cases for combinatorial testing. COMPSAC 2004. 2004; 72-77.
5. Ahmed BS, Sahib MA, Potrus MY. Generating combinatorial test cases using simplified swarm optimization (SSO) algorithm for automated GUI functional testing. Eng Sci Technol. 2014;17(4): 218-226.
6. Liang X, Guo S, Huang M, Jiao X. Combinatorial test case suite generation based on differential evolution algorithm. JSW. 2014; 9 (6): 1479-1484.
7. Rao R. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. Int J Ind Eng Comput. 2016; 7 (1): 19-34.
8. Singh SP, Prakash T, Singh V, Babu MG. Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm. Eng Appl Artif Intell. 2017;60: 35-44.
9. Warid W, Hizam H, Mariun N, Abdul-Wahab NI. Optimal power flow using the Jaya algorithm. Energies. 2016; 9 (9): 1-18.
10. Mishra S, Ray PK. Power quality improvement using photovoltaic fed DSTATCOM based on Jaya optimization. IEEE T Sustainable Energy. 2016; 7 (4): 1672-1680.
11. Othman RR, Zamli KZ. ITTDG: integrated t-way test data generation strategy for interaction testing. Sci Res Essays. 2011; 6 (17): 3638-3648.
12. Schroeder PJ, Faherty P, Korel B. Generating expected results for automated black-box testing. ASE 2002. 2002; 139-148.
13. Wang Z, Xu B, Nie C. Greedy heuristic algorithms to generate variable strength combinatorial test suite. QSIC 2008. 2008; 155-160.
14. Colbourn CJ, Cohen MB, Turban R. A deterministic density algorithm for pairwise interaction coverage. On IASTED Conf Softw Eng. 2004; 345-352.
15. Bryce RC, Colbourn CJ. A density-based greedy algorithm for higher strength covering arrays. Softw Test Verif Rel. 2009; 19 (1): 37-53.
16. Ong HY, Zamli KZ. Development of interaction test suite generation strategy with input-output mapping supports. Sci Res Essays, 2011, 6 (16): 3418-3430.
17. Mao C, Yu X, Chen J, Chen J. Generating test data for structural testing based on ant colony optimization. QSIC 2012, 2012, pp. 98-101.
18. Chen X, Gu Q, Li A, Chen D. Variable strength interaction testing with an ant colony system approach. APSEC 2009, 2009, pp. 160-167.
19. Ramli N, Othman RR Ali MSAR. Optimizing combinatorial input-output based relations testing using Ant Colony algorithm. ICED 2016, 2016, pp. 586-590.
20. Srivastava PR, Kim T. Application of genetic algorithm in software testing. Int J Softw Eng its Appl, 2009, 3 (4): 87-96.
21. McCaffrey JD. Generation of pairwise test sets using a genetic algorithm. COMPSAC 2009, 2009, 1, pp. 626-631.
22. Alsewari AA, Zamli KZ. Interaction test data generation using harmony search algorithm. In 2011 IEEE Symp Indu Elect Appli, Langkawi, Malaysia, 2011, IEEE Computer Society, pp. 559-564. DOI: 10.1109/ISIEA.2011.6108775.
23. Alsewari AA. A harmony search based pairwise sampling strategy for combinatorial testing. *International* J Phys Sci, 2012, 7 (7): 1062 - 1072.

24. Chen X, Gu Q, Qi J, Chen D. Applying particle swarm optimization to pairwise testing. COMPSAC 2010, 2010, pp. 107-116.
25. Ahmed BS, Zamli KZ. A variable strength interaction test suites generation strategy using particle swarm optimization. J Syst Softw, 2011, 84 (12): 2171-2185.
26. Cohen MB, Gibbons PB, Mugridge WB, Colbourn CJ. Constructing test suites for interaction testing. ICSE03, Portland, Oregon USA, IEEE Computer Society, 2003, pp. 38-48.
27. Cohen MB, Colbourn CJ, Ling ACH. Augmenting simulated annealing to build interaction test suites. ISSRE 2003, 2003, pp. 394-405.
28. Alsewari AA, Tairan NM, Zamli KZ. Survey on input output relation based combination test data generation strategies. ARPN J Eng Appl Sci, 2015, 10 (18): 8427-8430.
29. Arshem J. TVG. Available: http://sourceforge.net/projects/tvg (accessed on 1 June 2019).
30. Ziyuan W, Changhai N, Baowen X. Generating combinatorial test suite for interaction relationship. SOQUA 2007, ACM, Dubrovnik, Croatia, 2007, pp. 55-61.
31. Schroeder PJ, Korel B. Black-box test reduction using input-output analysis. ACM SIGSOFT, 2000, 25 (5): 173-177.

استراتيجية اختبار التوافقية القائمة على المدخلات والمخرجات باستخدام خوارزمية جايا

محمد عصام يونس[1]* عبد الرحمن أحمد السيواري[2] أن جي يونج خانج[2] كمال زهيري زاملي[2]

[1]قسم هندسة الحاسبات، كلية الهندسة، جامعة بغداد، بغداد، العراق
[2]كلية نظم الحاسوب ، جامعة ماليزيا باهانج ، باهانج ، ماليزيا

الخلاصة:

ويكاد يكون من المستحيل اختبار كل مجموعة من المدخلات نظرًا لأن تنفيذ حالات الاختبار يتطلب وقتا طويلا للغاية. الأختبار الاندماجي هو السبيل لتخطي عقبات الاختبار الشامل من خلال أختبار كل قيم المدخلات لكل المعاملات المركبة المتعددة طرق الترتيب. يمكن تقسيم الاختبار التجميعي إلى ثلاثة أنواع هي تفاعل القوة الموحد ، والتفاعل المتغير والقوة ، والعلاقة القائمة على المدخلات والمخرجات . ان الطريقة الاخيرة الانفة الذكر تختزل الفحص الاندماجي الى مجموعة ضمن اختيار الشخص الفاحص. معظم الابحاث في الاختبار الاندماجي طبقت في تفاعل القوة الموحدة وقوة التفاعل المتغيرة ، ومع ذلك ، هناك اهتمام قليل جدا بالعلاقة بين المدخلات والمخرجات. لذا تم اقتراح خوارزمية جايا في هذا البحث كخوارزمية مثلي لانشاء جدول الفحص الاندماجي باستراتيجية تعتمد على العلاقة بين المدخلات والمخرجات. نتيجة تطبيق خوارزمية جايا في الاختبار الاندماجي القائم على المدخلات والمخرجات مقبولة لأنها تنتج العدد الأمثل تقريبًا لحالات الاختبار في نطاق زمني مقبول.

الكلمات المفتاحية: خوارزمية جايا ، اختبار البرمجيات ، الاختبار الأندماجي ، اختبار متعدد الترتيب، موثوقية النظام.