International Journal of Engineering & Technology, 7 (3.8) (2018) 22-28

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Critical Review on Automated Test Case Generation for Conducting Combinatorial Testing Using Particle Swarm Optimization

Dr. V. Chandra Prakash[1], Subhash Tatale[2] *, Vrushali Kondhalkar[3] , Laxmi Bewoor[4]

[1]professor, [2,3,4] Research Scholar,
Department Of Cse, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Ap
**Corresponding Author E-Mail: Subhashtatale@Gmail.Com*

Abstract

In software development life cycle, testing plays the significant role to verify requirement specification, analysis, design, coding and to estimate the reliability of software system. A test manager can write a set of test cases manually for the smaller software systems. However, for the extensive software system, normally the size of test suite is large, and the test suite is prone to an error committed like omissions of important test cases, duplication of some test cases and contradicting test cases etc. When test cases are generated automatically by a tool in an intelligent way, test errors can be eliminated. In addition, it is even possible to reduce the size of test suite and thereby to decrease the cost & time of software testing.
It is a challenging job to reduce test suite size. When there are interacting inputs of Software under Test (SUT), combinatorial testing is highly essential to ensure higher reliability from 72 % to 91 % or even more than that. A meta-heuristic algorithm like Particle Swarm Optimization (PSO) solves optimization problem of automated combinatorial test case generation. Many authors have contributed in the field of combinatorial test case generation using PSO algorithms.
We have reviewed some important research papers on automated test case generation for combinatorial testing using PSO. This paper provides a critical review of use of PSO and its variants for solving the classical optimization problem of automatic test case generation for conducting combinatorial testing.

Keywords: *Particle Swarm Optimization (PSO); Software Testing; Combinatorial Testing; Pair-wise Testing; Automated Test Case Generation*

1. Introduction

Software testing is an important process of software development life cycle. However, software testing is costly while it gets concerned with human efforts, cost and time. The size of software increased enormously in the last 25 years. In the olden days, there was scarcely any software application that is more than 20K (Lines LOC. Now days, it is easy to find software application which has more than a million LOC. Such a remarkable rise in size of software has posed many problems in software testing.

In research and industrial communities, more emphasis is given by researchers and practitioners to discover automatic software testing techniques to ensure cost-effective and high-quality software delivery to customers or end users. At present, research in software testing area is chiefly concerned with problems like automatic test case generation, test coverage criterion design, fault localization, test oracle and regression testing. Among these problems, test case generation problem is considered as an essential problem in the research area of software testing. In spite of various input test generation techniques, combinatorial testing received major attention by the researchers [1].

In the next subsection, we have discussed the background of Combinatorial Testing and Particle Swarm Optimization.

1.1 Combinatorial Testing

This section provides essential background about the combinatorial testing. Test cases are generated using combinatorial testing according to SUT. Parameters and values describe an input to the system after analyzing the requirements. In order to cover each possible combination with the given parameters, a set of tests are generated. However, as all these parameter combinations, the generated test suite size becomes too large, and test suite cost becomes more expensive. To overcome such limitations, Cohen et al. [1] initially introduced the combinatorial testing concept and proposed a greedy strategy called Automatic Efficient Test Generator (AETG). This testing technique reduced the number of test cases to the extent by combinations of a few parameters. They demonstrated that the optimal test suite size increases with the number of parameters logarithmically. Pairwise testing is used to cover all the combinations of any two random parameters. Many t-way strategies (i.e., t indicates the interaction strength) are even used to cover the necessary interaction strength among the parameters.

Combinatorial test case generation considered as the NP-complete problem [1]. So, generating optimum number of test cases is a difficult task. At present, many approaches and tools have been proposed by researchers to explore near optimal combinatorial test suites.

D. Richard Kuhn et al. [4] states that some combinations of input domain never occur in practice. Richard Kuhn et al. [3] suggested that few faults which were generated because of unusual combinations. Hence, it has been concluded that influencing parameters on pairwise testing was based on software faults. Pair-wise combinatorial testing is an efficient and effective method to reduce the test suite size and detect about 70% to more than 90% of software faults [3].

Given an input as per in Table 1, input has three parameters (namely A, B, C) and two parameters have two values, and one parameter has three values. Total 3*2*2 = 12 test cases are required to test combinations of all these factors.

Table 1: Typical test input

A	B	C
3	1	5
10	2	7
5	-	-

The optimal test suite has only six tests by using pairwise testing as shown in Table 2.

Combinatorial testing is not able to cover all the parameter combinations, but it shows noticeable results to detect maximum faults in very less test suite size [1]. In table 2, the test suite size reduced from 12 to 6. It may not be fascinating, but we can check the effectiveness by considering a more complicated test input. Consider a test input which has 20 parameters and each parameter has ten values. By the way of exhaustive testing, 1020 test cases generated. Through the use of pairwise testing, we can reduce the test suite size to 213 test cases.

Table 2: An optimal test suite (using pairwise strategy)

S.N.	**A**	**B**	**C**
1	3	1	5
2	3	2	7
3	10	1	5
4	10	2	7
5	5	1	5
6	5	2	7

The following section explains some basic terminologies about combinatorial testing.

1) Parameter and Value: - Given an input, it has k parameters P = {p1, p2, · · · , pk}. After applying the combinatorial test techniques, the ith parameter pi has vi valid values and we denote this valid value set as Vi = {v i,1, v i,2, · · · , v i,vi}.

2) t-way Combinatorial Coverage Criterion: - Test coverage criterion generates Test Requirement (TR) which consists of each combination of arbitrary t parameters. t denotes coverage (interaction) strength of combinatorial coverage criterion. It is called as pairwise coverage criterion when interaction coverage t=2. The fault detection ability becomes stronger as coverage strength increases.

3) Combinatorial Test Suite: - A test suite TS called as a combinatorial test suite if for every combination tr (tr ∈ TR), there is at least one test case tc (tc ∈ TS) such that t covers tr. TR represents Test Requirement. The combinatorial test suites are optimal when the combinatorial test suite size is the minimum for a given input. When all parameters have the same number of values, a combinatorial test suite can called as covering array.

4) Covering Arrays: - A Covering Array CA (M; k, n, v), is an M × n array on v symbols satisfying that the rows of each M × k sub-array cover all t-sets from the k columns at least once [1]. In some testing scenarios, parameters can have the different number of values. In such cases, Mixed Covering Arrays are used.

5) Mixed Covering Arrays: - A Mixed Covering Array MCA (M; k, n, (v1, v2, • • • , vk)), is an M × n array satisfying that each column i ($1 \leq i \leq p$) contains only levels belonging to the set Vi and the rows of each M × k sub-array cover all t-sets from the t columns at least once [1]. In Table 2, an MCA (M; k, n, (v1, v2, • • • , vk)) is written as MCA (9;2,3,22)

Generally, test cases are derived from input domain of the SUT in combinatorial testing. When the input domain size is bigger and the output domain size is much lesser, test cases derived from output domain is considered preferably. Test cases derived from output domain in systems such as safety-critical embedded systems which ensure maximum output combinations are tested in detail [9].

1.2 Particle Swarm Optimization

This section briefs about the Particle Swarm Optimization (PSO) metaheuristic technique which majorly focuses on global search. PSO is inspired by social behavior which was observed in a flock of birds or schools of fishes. PSO is used generally as an optimization algorithm. The idea was motivated by food searching technique adopted by birds. This algorithm works on the similar principle and finds the best solution in the given search space. In this algorithm, social sharing among individuals is maintained by members of an entire population. Each member in the search space is called as a particle, and the whole search space is called as a swarm. Each particle represents a single solution. Each particle has a velocity which helps to move in the multi-dimensional search space. The objective function evaluates the fitness value of each particle. The velocity of each particle provides direction to move towards an approximate solution for the given objective function. Kennedy and Eberhart, 1995 defined standard theory and procedure of PSO.

Initially, the particles are initialized with random positions, and later it explores the search space for finding a better solution by changing its position. An each particle adjusts its velocity at an every iteration to follow two best solutions.

Fig. 1 shows the general workflow of a PSO-algorithm. Each particle represents a potential solution by maintaining its current position and velocity. Along with their individual solution, personal best (pbest) and global best (gbest) is maintained by swarm. The velocity and position is updated iteratively for exploring solution space. The exploitation was carried out by identifying promising neighbors and is largely depends on pbest and gbest value. Convergence of swarm or maximum number of iterations decides termination criteria for this iterative process.

A certain update rules are applied to manipulate the velocity and position of the particle around the search space.

Equation (1) indicates the rule for the velocity updation of particle.

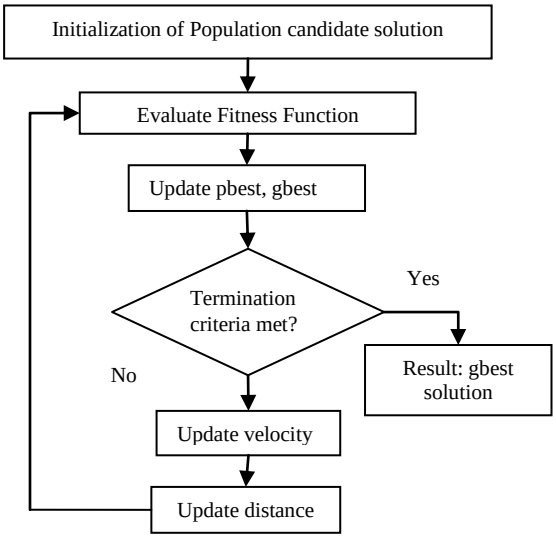

Fig. 1: Workflow of PSO Algorithm

$$V\text{-}MAX_{i,d}(n) = w\ V\text{-}MAX_{i,d}(n-1) + c1\ r_{i,d}(pbest_{i,d}(n-1) - X\text{-}MAX_{i,d}(n-1)) + c2\ r'_{i,d}(lbest_{i,d}(n-1) - X\text{-}MAX_{i,d}(n-1)) \quad .....(1)$$

Equation (2) indicates the rule for the position updation of particle.

$$X\text{-}MAX_{i,d} = X\text{-}MAX_{i,d}(n-1) + V\text{-}MAX_{i,d}(n) \quad .....(2)$$

Where i is particle index, d is the dimension, n is iteration number, w is the inertia weight factor having value between 0.4 and 1.4. r and r' are random numbers having value between 0 and 1. c1, c2 are acceleration coefficients that are used to adjust the weight between components. The acceleration constants c1 and c2 move towards target. If these values are low, it may roam far from actual target regions while high values will have speedy movements. Hence there is need to adjust acceleration coefficients appropriately. c1 is termed as cognitive parameter which determines bird's confidence and c2 is termed as social parameter which determines swarm's confidence. Normally, value of c1 is ranging from 1.5 to 2 and c2 is ranging from 2 to 2.5.

Based on the aforesaid rule, velocity of each particle is updated for better movement around the search space and this updated new velocity is used to find the new position of the particles.

The rest of this paper is organized as follows: Section 2 provides the critical review of PSO development in combinatorial test case generation. Section 3 briefs summary of reviews. Section 4 concludes this paper.

2. Literature Review

In this section, we have discussed the critical review of PSO variants in software test case generation for combinatorial testing. Application of PSO is observed in many combinatorial optimization problems. Bewoor et al. [29] compared the performance of PSO with Genetic Algorithm (GA) and Tabu Search (TS) and proved PSO is providing better solution as compared to other metaheuristic for stochastic input. Bewoor et al. [25] further developed Proposed Hybrid PSO (PHPSO) algorithm for solving combinatorial optimization problem of No Wait Flow Shop Scheduling (NWFSSP) and proved that PSO superior than other algorithms available in the literature. Inspired from the idea of applying PSO for NWFSSP this paper is an attempt for reviewing application of PSO for test case generation and following section provides a critical review on the same.

2.1 A Review on Combinatorial Test Case Generation

Pan et al. [10] [2011] introduced an organizational evolutionary particle swarm algorithm (OEPST). This algorithm is used to generate test cases for combinatorial testing. It adopted mainframe of AETG and combined characteristics of organizational evolution and PSO. This algorithm is used for selecting the test cases for converging towards local optimal solution. The method was focusing on the current environment. Test cases were identified from test suite based on pairwise coverage criterion. It ensured better retention of responsiveness, communication ability, memory, intelligent self-learning ability of particles. OEPST performed better when the number of parameter combinations is more significant. The experimental results showed that this algorithm efficiently reduced the test suite size.

Wang et al. [17] [2013] proposed an improved algorithm for test data generation based on PSO. Fitness value was used to show optimization degree of generated test data and to outline state for continue search. The number of current output test data was used dynamically according to earlier generated test data. Size of test data set generated by improved PSO algorithm was bit more significant than the test data set size by PSO. Improved PSO generated bit larger test data set than PSO, but spent less time to generate test data set. The authors claimed that efficiency of generated test suite increased on the ground of assuring the test suite optimization.

Akram Kalaee et al. [23] [2016] proposed optimal solution to generate test cases using Reduced Ordered Binary Decision Diagram (ROBDD) and PSO algorithm. These strategies are used to generate minimum test cases which achieve maximum coverage of input parameters. In order to meet the high quality of test suite ROBDD was proposed which uses the cause-effect graph. ROBDD represents compact and unique Boolean expressions invariably. The generated test suite size decreased due to significant effects considered from the graph. As the number of variables reduced without compromising testing quality, both testing cost and time also considerably reduced. A PSO algorithm was used to select test cases which cover all pairwise combinations of input parameters. The proposed approach achieved high efficiency for minimum optimized test suite size and maximum input parameter coverage.

Sahin et al. [24] [2016] used meta-heuristics techniques like Artificial Bee Colony (ABC), Differential Evolution Particle Swarm Optimization (DEPSO) and Firefly Algorithms (FA) and Random Search (RS) algorithm to explore test data. These algorithms were used to maximize coverage metric in search-based software testing and compared these algorithms from different fitness function. They suggested suitable values of control parameters as algorithms are heavily dependent on control parameters. They concluded that ABC algorithm has the least number of control parameters and does not need to fine-tune of parameters. Selecting optimal control parameters can produce excellent results for PSO and Differential Evolution (DE) algorithms which is an important task. Authors also commented that algorithms which are having more random nature were more successful. Meta-heuristics algorithms produced comparable and acceptable results for easy problem statements like even odd, remainder, largest, etc. ABC algorithm proved to be superior for problems that have multimodality (leap year problem statement) and many constraints (classification of triangle type and average marks problem). DE algorithm proved to be superior for triangle problem which required more neighbor information in search time. The performance of RS algorithm was worse compared to the meta-heuristics on the triangle and mark problems. RS algorithm was very efficient on easy problem statements in a small search space but was unsatisfactory on multimodality problems and large search space.

2.2 A Review on Test Case Generation Using Covering Arrays

Ahmed et al. [14] [2012] explained the application of PSO to construct uniform and t-way covering arrays. Authors explored the use of Artificial Intelligence (AI)-based algorithms like Simulated Annealing (SA), Ant Colony Algorithm (ACA), GA and TS as t-way (interaction strength) testing strategies. Many AI-based t-way testing strategies adopted complex search processes and required heavy computations. Because of this limitation, existing AI-based t-way testing strategies restricted to small interaction strength upto t= 3. There was a need to increase interaction strength up to t = 6 to detect maximum faults.

Authors proposed another effective and efficient strategy termed as Particle Swarm-based t-way Test Generator (PSTG). It is also used to construct uniform and variable strength covering arrays. PSTG adopted complex search processes with the lightweight computation. PSTG supported to high interaction strengths up to t = 6. Several experiments were conducted to evaluate the performance of PSTG. PSTG consistently performed comparatively better than AI and other existing t-way testing strategies to construct uniform and variable strength covering arrays.

Wu et al. [19] [2013] developed a new discrete PSO (DPSO). It adapted set-based PSO to generate optimized covering arrays. This strategy incorporates two additional approaches called particle re-initialization and further evaluation of gbest to improve the performance. Parameter tuning was applied to identify best parameter settings for conventional PSO and DPSO. The different types of PSO were implemented. The performance of these algorithms compared with other existing evolutionary algorithms like GA and ACO.

DPSO outperformed conventional PSO with minimum iteration. Their parameter settings considerably impacted performance of both conventional PSO and DPSO. Conventional PSO improved on previously reported results with recommended parameter settings. DPSO performed better than conventional PSO with recommended parameter settings [19].

Mahmoud et al. [21] [2015] proposed Fuzzy Interface system (FIS) designs for PSO parameter tuning. The objective to build proposed FISs is to monitor the performance of PSO approach and to overcome the optimization process problems by adjusting the parameters. This design improves efficiency.

Authors proposed an adaptive swarm optimization algorithm based on fuzzy logic. With compared to conventional methods, the current algorithm modified the heuristic parameters using a fuzzy-based self-adaptive approach. The results showed that PSO drastically improved its efficiency by adding the fuzzy-based adaptive mechanism for generated covering arrays size. However, additional computational requirements needed due to the addition of logic to the PSO. Thus, this strategy required more time to generate covering arrays with variable strength having more than four (t>4).

2.3 A Review on Pairwise Test Case Generation

Chen et al. [5] [2010] applied PSO to construct a single test. He proposed two algorithms based on one-test-at-a time strategy and Input Parameter Order (IPO)-like strategy to build final combinatorial test suites.

A methodology was developed and compared the performance of GA and ACA approach with 1000 iterations. Ten independent iterations performed and high statistical confidence observed. Three boundary handlings strategies were used namely Absorbing Walls, Reflecting Walls and Cyclic Walls. Out of these three different boundary handling strategies, Reflecting Walls strategy was the most effective strategy among these strategies. Absorbing Walls boundary handling strategy is used to complete a single test. From empirical study, authors found that IPO based PSO strategy is more efficient than One-test-at-a-time based PSO strategy. Authors also analyzed the effectiveness of increasing iteration number. Authors found that increasing iteration number generate smaller test suite size with more complicated inputs. However, it also noticed that increasing iteration number got the worse result with more complicated inputs.

SA always generated minimum test suites in all inputs. But due to many computational resources, SA approach was time-consuming. Current PSO based strategy still achieved promising results when input was more complicated.

The execution time, programming language and platform of different approaches considered verifying the efficiency and effectiveness of PSO approach. Based on this environment, authors claimed that meta-heuristic search techniques take considerable time compared to other greedy methods in most inputs.

Zha, Ri-Jun et al. [7] [2010] proposed statistics based Cross-Entropy method and PSO to generate pairwise test cases. The cross-entropy method used the best selection probability for test case generation. A theoretical method of the cross-entropy method considered for the optimal probability selection. Test cases generated by searching best fitness value from a feasible solution space. The size of test suite is reduced by these two methods. The empirical results showed that the cross-entropy method and the PSO have some excellent qualities compared with existing algebra methods, greedy algorithms, and other heuristic search algorithms.

Ahmed et al. [12] [2011] proposed Pairwise Particle Swarm based Test Generator (PPSTG) to generate pairwise test cases. The study highlighted PPSTG design and compared its performance regarding test size with other existing strategies. PPSTG strategy compared with published results from literature for evolutionary algorithms viz. GA, ACA, AETG, mAETG & IPO.

As PPSTG generates non-deterministic results, 20 independent iterations performed to ensure high statistical confidence. Best results reported with these independent iterations. PPSTG performed better than other existing tools like Test Vector Generator (TVG), Pairwise Independent Combinatorial Testing (PICT), Test Configurations (TConfig), IPOG, Jenny and Classification-Tree Editor Extended Logic (CTE-XL). The PPSTG strategy produced promising results regarding test suite size.

2.4 A Review on T-Way Test Case Generation

Ahmed et al. [6] [15] [2010] produced comparisons with some existing strategies and tools like WHITCH, FireEye, TConfig, Jenny, and TVG to evaluate the effectiveness in term of test suite size against other strategies. Proposed strategy compared with other strategies by performing different experiments by considering uniform and mixed variable parameters taken as a system configuration and the variable interaction strength.

High statistical confidence achieved by performing maximum independent iterations. Proposed PSTG strategy performed better than other strategies for generated test suite size and gave an optimal test suite size in most of the experiments [6]. TConfig, WHITCH, and PSTG produced the most optimum size of test suite for interaction strength (t) =3. WHITCH produced best results when interaction strength (t) =≤ 3 whereas it could not support when interaction strength (t)>4. TConfig produced best possible test cases when t=2 and t=6.

Ahmed et al. [11] [2011] explained Variable Strength (VS)-PSTG strategy for test suite generation. The strategy was combined with PSO algorithm for t-way to construct VS test suites. SA, TVG and ACA strategies produced non-deterministic results. Since these strategies are dependent on some degree of randomness. In this strategy, each configuration executed ten times and selects smaller test suite size as best test suite. However, for Density, PICT, IPOG, ITCH, and ParaOrder strategies produced deterministic results. Since these strategies do not depend on some degree of random-

ness. Only one cycle is sufficient for these strategies. Overall, VS-PSTG gave promising results [11].

Ahmed et al. [13] [2012] presented the design and implementation of PSTG to generate t-way test suite. Experiments divided into three parts like parameter tuning of the PSTG strategy, evaluation and comparison of PSTG with artificial intelligence-based strategy and comparison of PSTG with other computational-based strategies.

The relevant PSTG parameters tuning was essential to obtain the most optimal test size. These parameters consist of the acceleration coefficients, the inertia weight, the swarm size and the maximum iterations. The acceleration coefficients and inertia weight have a direct impact on the test sizes. It observed that PSTG achieved the best global solution when the acceleration coefficients were less than < 1 and best local solutions when inertia weight approaches to 0.9. It also concluded that an excellent choice for acceleration coefficients would be between 1.2 and 1.4 whereas the value of inertia weight would be between 0.2 and 0.5. It also analyzed that the performance of PSTG improved as increased the swarm size. For comparison with artificial intelligence-based strategies, test sizes generated by PSTG were very close to those of GA and ACA in most cases. PSTG also compared with other computational-based strategies like IPOG, WHITCH, Jenny, TConfig, and TVG. PSTG produced competitive results, and it competed with different strategies in most cases. Test sizes of PSTG were increased exponentially and logarithmically as the variable strength, parameters, and values increased.

Authors also demonstrated evaluation of lexical analyzer case study to compare the capability of PSTG in generating efficient test suites. PSTG efficiently generated the required test suites. The 2-way test suite detected 61.11 %; 3-way test suite detected 77.77 % of the seeded faults including three new ones which have not been detected by the 2-way test suite. The 4-way test suite detected 83.33 % of the seeded faults with only one additional fault, which was undetected by the 3-way test suite. Newly seeded faults have not been detected in the case of 5-way and 6-way test suites.

Chana et al. [16] [2012] proposed novel effective PSO strategy to construct optimal t-way interaction test suites over the cloud to achieve near-optimal solution. Proposed approach performed better when interaction strength (t) =6 [16].

Rabbi et al. [20] [2015] proposed Swarm Intelligent Test Generator (SITG) which is a test data generator inspired by PSO. The result shows that SITG is more acceptable as compared to other strategies like IPOG, WHITCH, Jenny, TConfig, and TVG. SITG can consider as the optimum test data generation strategy. The analysis also showed that improvement up to 5% is possible through SITG regarding the test suite size [20].

2.5 A Review on Combinatorial Test Case Generation for Graphical User Interface

Ahmed et al. [18] [2014] proposed strategy to generate combinatorial test cases for functional testing of GUI using Simplified Swarm Optimization (SSO) with the support of Event Interaction Graph (EIG). The problem with the conventional PSO was that the speed decreases as the number of iteration is increased, which affects the particles to achieve the best value. The SSO theory was used to optimize the test cases considering the combinatorial testing concepts to overcome the drawback of PSO. The SSO helped to generate better results regarding test size and time. A random event deleted which are not applicable to repairing and managing test suite. SSO discarded personal influence (cognitive) term in the velocity adjustment equation. They gave a valuable insight into the fact that combinatorial testing could be used efficiently and effectively for the various case studies. It generated small size test case suite while mostly time to generate test suite is better than original PSO version.

2.6 A Review on Constraints Handling In Combinatorial Testing

Ahmad et al. [26] [2017] proposed multi-objective particle swarm and multithreading techniques to handle constraints in combinatorial testing. This strategy is a combination of the new algorithms i.e., input parameter combination generation algorithm using a stack, hashing algorithm to store search space, multi-objective multithreaded PSO for constraints satisfaction and neighbored search to achieve better coverage. Parameter generation algorithm evaluation showed that through performance dropped when interaction strength increased still algorithm performed very well. This algorithm grows logarithmically not linearly for all values of t. Searching time of t-tuple in search space reduced dramatically and hence can reduced total generation time of the solution. Results of hashing based algorithm compared with two other algorithms, indexing search, and full search. Indexing search performed better than full search, but its performance dropped with an increase in interaction strength. Hashing based search performed better. Looking at generating efficiency of this strategy it observed that this outperformed AETG and mAETG because they are using one row at a time approach.

Sheng et al. [27] [2017] proposed constraints test case generation for combinatorial testing using PSO. PSO Constraints Test cases Generation algorithm (PCTG) with two variations. PCTG-Av avoids the selection of conflicting test cases and uses fitness function to choose the best and valid particle gBest after each cycle. PCTG-Re selects the best particle gBest using fitness function and then handles the constraints by replacing different test case. It also provided an optimal match of the factor which can lead PSO to move to optimal position. The PCTG-Av selects the test case which has the most significant fitness factor and satisfies the constraint validity as the gBest. PCTG-Re chooses the test case which only has the most significant fitness factor as gBest during the iteration process. After the iteration process, if the gBest does not satisfy the constraint validity, the strategy of replacing is applied to update it. PCTG-Re achieves good result than the PCTG-Av and defeats other strategies in most cases as long as sizes of the generated constraints covering array are concerned.

3. Summary of Literature Review

In this section, a summary of PSO variants used in test case generation for combinatorial testing is explained. Table I briefs about summary of literature review. Many authors have contributed in the research area of combinatorial testing.

Table 1: A Summary of Literature Review

S.N.	Authors	Problem discussed and solved	Method/ Algorithm/ Tools Used	Results
1	Chen et al. [5]	To generate test cases in pairwise testing	one-test-at-a-time strategy & IPO	It runs faster than GA, ACA, SA and AETG.
2	Ahmed et al. [6][15]	To generate test cases using t-way strategy	PSTG strategy	PSTG produced the most optimum test suite size than WHITCH, TConfig.
3	Zha, Ri-Jun et al. [7]	To generate combinatorial test cases and comparison based on cross-entropy and PSO method	cross-entropy and PSO method	The cross-entropy method and PSO performed better than existing greedy algorithms, algebra methods and other heuristic search algorithms.

4	Ahmed et al. [9][13]	To generate test cases using t-way strategy	PSO	Generates comparable results with respect to test suite size.
5	Pan et al. [10]	To generate combinatorial test cases	OEPST	It ensures better retention of responsiveness, communication ability, memory, collaboration and intelligent self-learning ability of particles. OEPST performs better when number of parameter combinations is larger.
6	Ahmed et al. [11][14]	To generate t-way interaction test suite	VS-PSTG strategy	Produces non-deterministic & promising results.
7	Ahmed et al. [12]	To generate test cases in pair-wise testing	PPSTG algorithm	Ensures high statistical confidence. Generates comparable better results with respect to test suite size than PICT, TConfig, IPOG, Jenny, CTE-XL.
8	Chana et al.[16]	To construct optimal t-way interaction test suites over cloud	PSO	Proposed approach performed better when interaction strength (t) =6.
9	Wang et al. [17]	To generate combinatorial test cases	PSO, Improved PSO algorithm	An improved PSO algorithm is effective and efficient for complex systems in terms of time spent.
10	Ahmed et al. [18]	To generate combinatorial test cases for automated GUI functional testing	PSO, SSO,CA	Produces knowledge based rules. Controls optimization performance
11	Wu et al. [19]	To generate covering arrays	PSO, DPSO	DPSO performed better than conventional PSO with less iteration, and the performance of conventional PSO and DPSO was significantly impacted by their parameter settings.
12	Rabbi et al. [20]	To generate test cases using t-way strategy	PSO, SITG	An improvement up to 5% is possible through SITG in terms of the number of test data set. SITG is more acceptable than IPOG, WHITCH, Jenny, TConfig and TVG.
13	Mahmoud et al. [21]	To construct covering arrays	Fuzzy logic, APSO	Improved efficiency in terms of the size of generated arrays. Optimize the number of test cases effectively.
14	Kalaee et al. [24]	To generate combinatorial test cases	ROBDD Graph and PSO Algorithm	Proposed approach achieves high effectiveness. It reduces testing cost by selecting appropriate test cases.
15	Sahin et al.[25]	To generate combinatorial test cases	Metaheuristic Algorithm	For easy problems meta-heuristics produce comparable and acceptable results. For multi-modality problems and a number of constraints ABC proves to be superior.
16	Ahmed et al. [27]	To generate constraints test cases in combinatorial testing	MOPSO	Produced impressive results with binomial time for generation.
17	Sheng et al. [28]	To generate test cases for handling constraints in Combinatorial Interaction Testing (CIT)	PCTG-Av, PCTG-Re	PCTG-Re performs better than the PCTG-Av and outperforms other strategies in most cases as far as the generated constraints covering array sizes are concerned.

4. Conclusion

This paper provides a critical review of automatic test case generation for conducting combinatorial testing using different types of PSO techniques. This paper covers the review of 23 research papers. As the number is very less, it indicates that a lot more research paper can be published in this research area in future. It is found that PSO can be used efficiently to generate an optimized number of test cases for conducting combinatorial testing.

References

[1] Cohen, David M., Siddhartha R. Dalal, Michael L. Fredman, and Gardner C. Patton. "The AETG system: An approach to testing based on combinatorial design." IEEE Transactions on Software Engineering Vol.23, No. 7 (1997), pp. 437-444.

[2] Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization." Swarm intelligence, Springer, Vol. 1, No. 1 (2007), pp. 33-57.

[3] R. Kuhn, Yu Lei and Raghu Kacker, "Practical Combinatorial Testing: beyond Pair wise", IEEE Computer Society - IT Professional, Vol. 10, No. 3 (2008).

[4] D. Richard Kuhn, Raghu N. Kacker and Yu Lei, "Practical combinatorial testing", NIST Special Publication, (2010).

[5] Chen, Xiang, Qing Gu, Jingxian Qi, and Daoxu Chen. "Applying particle swarm optimization to pairwise testing." Conference In Computer Software and Applications (COMPSAC), IEEE (2010), pp. 107-116.

[6] Ahmed, Bestoun S., and Kamal Z. Zamli. "PSTG: a t-way strategy adopting particle swarm optimization." 4th Asia International Conference In Mathematical/Analytical Modelling and Computer Simulation (AMS), IEEE (2010), pp. 1-5.

[7] Zha, Ri-Jun, De-Ping Zhang, Chang-Hai Nie, and Bao-Wen Xu. "Test data generation algorithms of combinatorial testing and comparison based on cross-entropy and particle swarm optimization method." Jisuanji Xuebao (Chinese Journal of Computers) Vol.33, No. 10 (2010), pp.1896-1908.

[8] Ahmed, Bestoun S., and Kamal Z. Zamli. "T-way test data generation strategy based on particle swarm optimization." 2nd International Conference In Computer Research and Development, IEEE (2010), pp. 93-97.

[9] Vudatha, Chandra Prakash, Sateesh Nalliboena, Sastry Kr Jammalamadaka, Bala Krishna Kamesh Duvvuri, and L. S. S. Reddy. "Automated generation of test cases from output domain of an embedded system using Genetic algorithms." 3rd International In Electronics Computer Technology (ICECT), IEEE (2011), vol. 5, pp. 216-220.

[10] Pan, Xiaoying, and Hao Chen. "Using organizational evolutionary particle swarm techniques to generate test cases for combinatorial testing." 7th International Conference In Computational Intelligence and Security (CIS), IEEE (2011), pp. 1580-1583.

[11] Ahmed, Bestoun S., and Kamal Z. Zamli. "A variable strength interaction test suites generation strategy using Particle Swarm Optimization." Journal of Systems and Software Vol.84, No. 12 (2011), pp. 2171-2185.

[12] Ahmed, Bestoun S., Kamal Z. Zamli, and C. Lim. "The development of a particle swarm based optimization strategy for pairwise testing." Journal of Artificial Intelligence Vol.4, No. 2 (2011), pp. 156-165.

[13] Ahmed, Bestoun S., Kamal Z. Zamli, and Chee Peng Lim. "Constructing a t-way interaction test suite using the particle swarm optimization approach." International Journal of Innovative Computing, Information and Control Vol.8, No. 1 (2012),pp. 431-452.
[14] Ahmed, Bestoun S., Kamal Z. Zamli, and Chee Peng Lim. "Application of particle swarm optimization to uniform and variable strength covering array construction." Applied Soft Computing Vol. 12, No. 4 (2012), pp. 1330-1347.
[15] Ahmed, Bestoun S., and Kamal Z. Zamli. "A greedy particle swarm optimization strategy for t-way software testing." Journal of Artificial Intelligence Vol.5, No. 2 (2012), pp. 85-90.
[16] Chana, Inderveer, and Ajay Rana. "An Effective Approach to Build Optimal T-way Interaction Test Suites over Cloud Using Particle Swarm Optimization." In International Conference on Advances in Communication, Network, and Computing, Springer (2012), pp. 193-198.
[17] Wang, Jianfeng, Chao Sun, and Shouda Jiang "Improved algorithm for combinatorial test data generation based on particle swarm optimization." Journal of Harbin Engineering University Vol.4 (2013).
[18] Ahmed, Bestoun S., Mouayad A. Sahib, and Moayad Y. Potrus. "Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing." International Journal of Engineering Science and Technology, Vol. 17, No. 4 (2014), pp.218-226.
[19] Wu, Huayao, Changhai Nie, Fei-Ching Kuo, Hareton Leung, and Charles J. Colbourn. "A discrete particle swarm optimization for covering array generation." IEEE Transactions on Evolutionary Computation Vol.19, No. 4 (2013), pp. 575-591.
[20] Rabbi, Khandakar, Quazi Mamun, and MD Rafiqul Islam. "An efficient particle swarm intelligence based strategy to generate optimum test data in t-way testing." 10th Conference In Industrial Electronics and Applications (ICIEA), IEEE (2015), pp. 123-128.
[21] Mahmoud, Thair, and Bestoun S. Ahmed. "An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use." Expert Systems with Applications Vol.42, No. 22 (2015), pp. 8753-8765.
[22] V.Chandra Prakash and Kadiyala Priyanka, 2016. "Test Case Generation for Pairwise + Testing." Asian Journal of Information Technology. Vol. 15 No.23 (2016), pp.4800-4805.
[23] Kalaee, Akram, and Vahid Rafe. "An Optimal Solution for Test Case Generation Using ROBDD Graph and PSO Algorithm." International Journal of Quality and Reliability Engineering Vol.32, No. 7 (2016), pp. 2263-2279
[24] Sahin, Omur, and Bahriye Akay. "Comparisons of metaheuristic algorithms and fitness functions on software test data generation." Applied Soft Computing Vol. 49 (2016), pp. 1202-1214.
[25] Bewoor, L., V. Chandra Prakash, and Sagar U. Sapkal. "Comparative analysis of metaheuristic approaches for m-machine no-wait flow shop scheduling for minimizing total flow time with stochastic input." International Journal of Engineering & Technology, Vol.8 (2016), pp. 3021-3026.
[26] Ahmed, Bestoun S., Luca M. Gambardella, Wasif Afzal, and Kamal Z. Zamli. "Handling constraints in combinatorial interaction testing in the presence of multi objective particle swarm and multi-threading." Journal of Information and Software Technology Vol.86 (2017), pp. 20-36
[27] Sheng, Yunlong, Changan Wei, and Shouda Jiang. "Constraint Test Cases Generation Based on Particle Swarm Optimization." International Journal of Reliability, Quality and Safety Engineering Vol.24, No. 05 (2017).
[28] Bewoor, Laxmi A., V. Chandra Prakash, and Sagar U. Sapkal. "Comparative Analysis of Metaheuristic Approaches for Makespan Minimization for No Wait Flow Shop Scheduling Problem." International Journal of Electrical and Computer Engineering Vol.7, No. 1 (2017), pp. 417.
[29] Bewoor Laxmi A., V. Chandra Prakash, and Sagar U. Sapkal. "Evolutionary Hybrid Particle Swarm Optimization Algorithm for Solving NP-Hard No-Wait Flow Shop Scheduling Problems." Algorithms Vol.10, No. 4 (2017), pp. 121.
[30] Vudatha Chandra Prakash, Sastry K R Jammalamadaka, and Bala Krishna Kamesh Duvvuri. "Automated generation of Test cases for testing critical regions of embedded systems through Adjacent Pairwise Testing." International Journal of Mathematics and Computational Methods in Science & Technology Vol.2, No.2, (2012), pp. 10-15.
[31] Dr. Chandra Prakash V, Dr. Sastry J K R, Sravani G, Manasa USL, Khyathi A, Harini A. "Testing Software Through Genetic Algorithms-A Survey" Journal of Advanced Research in Dynamical and Control Systems, Vol. 9 (2017).