International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

Hybrid Particle Swarm and Ranked Firefly Metaheuristic Optimization-Based Software Test Case Minimization

M. Bharathi, Government Arts and Science College, Pennagaram, India

ABSTRACT

Software testing is a valuable and time-consuming activity that aims to improve the software quality. Due to its significance, combinatorial testing focuses on fault identification by the interaction of a small amount of input factors. But, deep testing is not sufficient due to time or resource availability. To select the optimal test cases with least computation time, hybrid multi-criteria particle swarm and ranked firefly metaheuristic optimization (HMCPW-RFMO) technique is introduced. Initially, the population of the test cases is randomly initialized. Then the fitness is calculated by the pairwise coverage, execution cost, fault detection capability, and average execution frequency. RFM approach starts with 'n' fireflies. The light intensity of each firefly gets initialized. If the light intensity of one firefly is smaller than the other one, it moves near the brighter one. Next, the rank is given to the firefly based on their light intensity. Lastly, the high ranked firefly is chosen as a global best solution. The result reveals that HMCPW-RFMO technique improves the software quality.

KEYWORDS

Combinatorial Testing, Light Intensity, Multi-Criteria Test Case Selection, Particle Swarm Optimization, Ranked Firefly Metaheuristic Optimization, Software Testing, Test Cases

1.0 INTRODUCTION

With Software testing is one of the significant task of software development. software testing is crucial to check the software application program to promise that the system is working accurately. Software testing is the momentous part of an effective software product. Software testing is the way of executing the software program for locating faults which trigger software failure. Most important objective of software testing is to recognize faults exist in software under development. Combinatorial testing (CT) is a valuable software testing method used to recognize the faults in couple of feature mixtures of Software Program Applications (SPA). Software testing demands time and cost consumed on software development. This cost may reduce quickly as testing time reduces. Software may be published into the market without being tested adequately due to marketing force and idea to save time and reduce costs. Publishing software products without high quality into the market is unacceptable. Because it may trigger loss of incomes or even damage of life. Accordingly, software testers should construct high-quality test cases that discover maximum of the faults in the software within the scheduled time for testing. Excellence of software program is attained by means of accurate test suite. CT is crucial for effectual test suite generation. CT facilitates the tester to perform testing with a small set of test cases for achieving very high fault coverage. Essential goal of software testing is to create a set of tiniest test cases which covers higher faults in smaller amount of time.

DOI: 10.4018/IJAMC.290534

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

Therefore, the excellence of software program is determined in terms of software metrics like highest faults type coverage, tiniest test suite size and least computational time. So, test suite minimization methods perform a most important role in minimizing the number of test suites and reducing time and cost of software testing without modifying their quality. Numerous research works have been performed for test case reduction to enhance the quality of software system. Still, time complexity of software testing was not reduced. Hence, this review work aims test suite optimization effectively for enhancing the capability of Software.

We have identified two different heuristic algorithms called (Bestoun S. Ahmed, 2016) Cuckoo Search Algorithm (CSA) and (Annibale Panichella et al., 2015) Diversity based Genetic Algorithm (DIV-GA) as baseline algorithms for performing this research.A cuckoo search algorithm (CSA) was designed in (Bestoun S. Ahmed, 2016) with the combinatorial method for reducing the test case and improving the fault detection. The CSA detects the configuration-aware faults but it does not considerably improve the performance of test case minimization. A Diversity based Genetic Algorithm (DIV-GA) was introduced in (Annibale Panichella et al., 2015) for selecting the test case with multiple objectives. The DIV-GA display more faults at a similar level of execution cost. The algorithm does not minimize the testing cost at a required level. But we identified that both baseline algorithms are demanded less fault coverage capability, lack of test case minimization, failure to solve the multi-objective problems, failure to improve software quality, more computational time to optimize the test suite.

To solve the above quoted issues in test suite optimization problem using cuckoo search algorithm and Diversity based Genetic Algorithm, Hybrid Multi-criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization (HMCPW-RFMO) is developed. HMCPW-RFMO algorithm is built based on higher faults-type coverage analysis and produce an efficient optimum results. The major contribution of the proposed HMCPW-RFMO technique is described as follows,

Ø To enrich the capability of test suite reduction for software testing.
Ø To enlarge the test suite reduction rate and to cut the execution time for software testing compared with cuckoo search algorithm and Diversity based Genetic Algorithm .
Ø To optimize test cases in test suite in terms of higher faults type coverage and less execution time.

The rest of the paper is organized as follows. The works related to optimal test case reduction and selections are discussed in Section 2. Section 3 provides the detail explanation of the proposed HMCPW-RFMO technique with neat diagram. Section 4 provides the experimental evaluation with the required dataset. The experimental results and discussion of the proposed and existing methods are described in section 5 with the number of parameters. Finally, the conclusion of the research work is presented in section 6.

2.0 RELATED WORKS

A hybrid intelligent algorithm called the branch and bound along with hill climbing was designed in (Ying Xing et al., 2014) for generating the test data. This hybrid algorithm failed to generate optimal test data for obtaining high coverage rate. An optimal software test suite selection was the significant problems in software engineering research. Artificial Bee Colony based novel search technique was designed in (Soma Sekhara Babu Lam et al., 2012) for automatic generation of software test suites. A novel one-test-at-a-time algorithm was developed in (Zhiqiang Zhang et al., 2014) for combinatorial test generation using pseudo-Boolean optimization. The algorithm does not improve the test generation since it does not use any heuristic knowledge algorithm. A simulated annealing and genetic algorithm approach were developed in (Justyna Petke et al., 2015) for the restricted combinatorial interaction testing with test suite generation. The approach failed to improve both coverage and fault detection.

A hyper-heuristic selection and acceptance method was introduced in (Kamal Z.Zamli et al., 2017) for generating the combinatorial t-way test suite with various optimization technique. The mechanism does not solve the dynamic multi-objective problems in the combinatorial t-way test suite generation. A novel multilevel optimization framework for bipartite networks was introduced in (Cemal Yilmaz, 2013). The framework does not handle combinatorial optimization problems in large bipartite networks.

A novel algorithm named complete Fault Interaction Location was introduced in (Wei Zheng et al., 2016) for minimizing the fault interaction set of test. The algorithm failed to cover more faults since it does not use any optimization approach. A Model-Based Test Generation Architecture was designed in (Saurabh Karsoliya et al., 2013) for reducing the test cases based on the testing criteria and requirements. The architecture also minimizes the pairwise coverage problem but it does not effectively provide maximum coverage. An Annealed algorithm and differential evolution algorithm were introduced in (Yuexing Wang et al., 2018) to create cost-aware mixed covering arrays for combinatorial testing. The algorithm does not find the optimal test cases for improving the coverage capability. A feedback driven adaptive combinatorial testing process was introduced in (Cemal Yilmaz et al., 2014) for detecting and locating failures in the software programs. The approach failed to use any optimization algorithm for selecting the best test cases. Metaheuristic approach called simulated annealing algorithm was developed in (Himer Avila-George et al., 2013) for constructing an efficient test-suites. But the algorithm does not improve the coverage rate with these generated optimal test cases. Cuckoo Search approach was designed in (Bestoun S. Ahmed et al., 2015) for combinatorial test suite generation. Though the approach detecting the more faults with optimal tests suits, the computation time for optimal test suite generation was high. A hybrid PSO and harmony search algorithm was introduced in (Luciano Soares de Souza et al., 2015) for optimal test cases selection with multiple objectives. The algorithm failed to get the better fault coverage rate with the selected test cases.

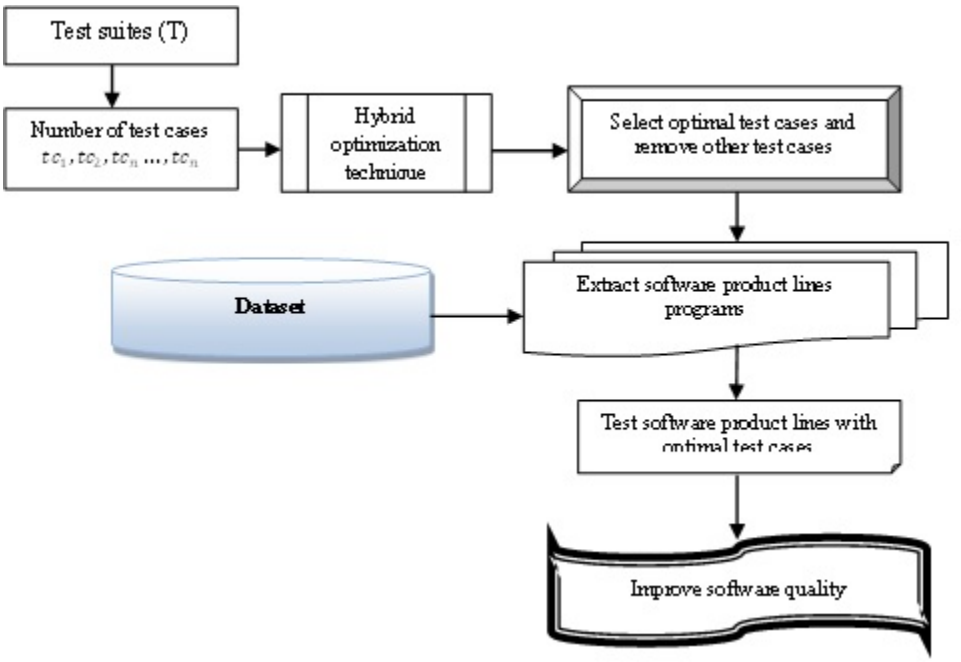
A Simplified Swarm Optimization approach was introduced in (Bestoun S. Ahmed et al., 2014) to produce an optimized test suite using Event-Interaction Graph. But the optimization performance was not improved since it failed to use the hybrid technique. In (Haider et al., 2014), a fuzzy-based optimization method was developed for multiple objective test suite selection. Though the method minimizes the test cases, it takes more time to select optimal test suite. A fuzzy logic-based adaptive swarm optimization technique was introduced in (Thair Mahmoud & Bestoun S.Ahmed, 2015) for software combinatorial testing process. The performance of optimization was not improved considerably by designing an efficient data structure to investigate faster for the particular combination. A pairwise testing with constraints algorithm was designed in (Xiao-Fang Qi et al., 2017) for detecting the more faults in a given amount of time. The algorithm failed to avoid irrelevant test suite for minimizing the complexity since it does not use any heuristic strategies. A hybrid variant of the flower pollination algorithm was introduced in (Abdullah B. Nasser et al., 2018) for generating the new t-way test suites. The algorithm fails to generate global best results.

3.0 HYBRID MULTI-CRITERIA PARTICLE SWARM AND RANKED FIREFLY METAHEURISTIC OPTIMIZATION BASED TEST SUITE REDUCTION

A software testing is an important process to guarantee the quality of the software system. Combinatorial testing (CT) is an efficient method to test the software with several configurable parameters. It is also used to discover the faults caused by the grouping effect of parameters. The combinatorial software testing is performed through the number of test cases. The test case is a set of conditions to satisfy the user requirements. The number of test cases collectively called as test suites. It is otherwise called the test suite comprises the number of test cases. While utilizing a large number of test cases, the system takes more time for testing the softwares. As a result, the quality of software gets minimized. In order to overcome such kind of issues, an efficient technique is required. Metaheuristics can discover best solutions among a large set of feasible solutions with less computational time than optimization

algorithms for combinatorial optimization problems. So, Hybrid Multi-Criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization technique is introduced for an optimal test case selection and reduction which improves the software quality. The general particle swarm optimization technique is a stochastic optimization technique based on the movement of the swarms (i.e. particles) around the search space. This optimization does not find a most optimal solution while considering the multiple criterions. This difficulty is overcome by the HMCPW-RFMO technique. The particle swarm optimization combined with the Ranked Firefly Metaheuristic approach effectively finds the global best solution through the ranking approach concerned with more than one objective function. The HMCPW-RFMO technique uses the multi-criteria optimization where the optimal decisions need to be taken in the presence of trade-offs between two or more unpredictable objectives. Therefore, the hybrid technique obtains a global optimization solution in the test case selection for combinatorial testing. The HMCPW-RFMO technique uses general particle swarm optimization technique to find local best solutions and ranked firefly metaheuristic approach to find global best solutions. The architecture of the HMCPW-RFMO technique is illustrated in Figure 1.

Figure 1. Architecture of HMCPW-RFMO technique



As shown in Fig 1, the architecture of HMCPW-RFMO technique is illustrated to improve the software system quality. The software quality improvement is attained by selecting the optimal test cases. Let us consider $T = \left\{ tc_1, tc_2, tc_n \ldots, tc_n \right\}$ where T denotes a test suite and tc denotes test cases. In order to optimize the test cases for combinatorial testing, hybrid optimization technique called HMCPW-RFMO is developed. The HMCPW-RFMO technique selects the optimal test cases among the number of test cases in the test suite. After that, the HMCPW-RFMO technique tests the number of software product lines extracted from the dataset with the selected optimal test cases which satisfying the user requirements. The selected test cases are also improves the fault coverage capability for testing the software programs. This aids to improve software quality.

3.1 Hybrid Multi-Criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization (HMCPW-RFMO)

A new Multi-criteria optimization algorithm is introduced for solving the problem of optimal test case selection with multiple criteria. In contrast to one objective problem, Multi-criteria optimization aims to optimize more than one objective functions at the same time. In order to use conventional optimization technique for test case selection, the global optimum solution was not obtained since it failed to satisfy the multiple criteria at the same time. This problem is overcome by introducing a new hybrid technique by combining the ranked firefly metaheuristic approach into the binary Multi-criteria particle swarm optimization for optimal test case selection. Test suite minimization is essential in combinatorial testing to improve pairwise coverage and to reduce the number of test cases.

- The HMCPW-RFMO technique integrates the ranked firefly metaheuristic combined into Multi-criteria Particle Swarm optimization to solve multi-criteria test case selection problems for combinatorial software testing process. The Particle Swarm optimization initializes the number of particles (i.e. test cases) and their positions. The fitness of each particle is calculated with multi-criteria such as pairwise coverage, execution cost, fault detection capability, fault tolerance capability and average execution frequency. If the fitness is better than the Pbset, then the current test cases are selected as local best.
- To find global optimal test cases, Ranked Firefly Metaheuristic approach is implemented with the Particle Swarm Optimization. The light intensity of each firefly is formulated with the fitness function. The fireflies are ranked based on their light intensity. The firefly with high rank is selected as global best test cases. Other test cases are removed. This helps to improve the test case reduction rate and minimizes the computation time.
- The software product lines are tested with the global best test cases to improve the fault coverage rate and minimize the testing cost.

Flow process of HMCPW-RFMO technique is illustrated in Fig 2.

Figure 2 shows the flow process of the global optimal test case selection using a hybrid optimization technique. The particle swarm is a population-based search procedure. In this approach, the number of particles (i.e. test cases) and their positions are initialized in search space. HMCPW-RFMO technique considers a Multi-criteria optimization problem is formulated as,

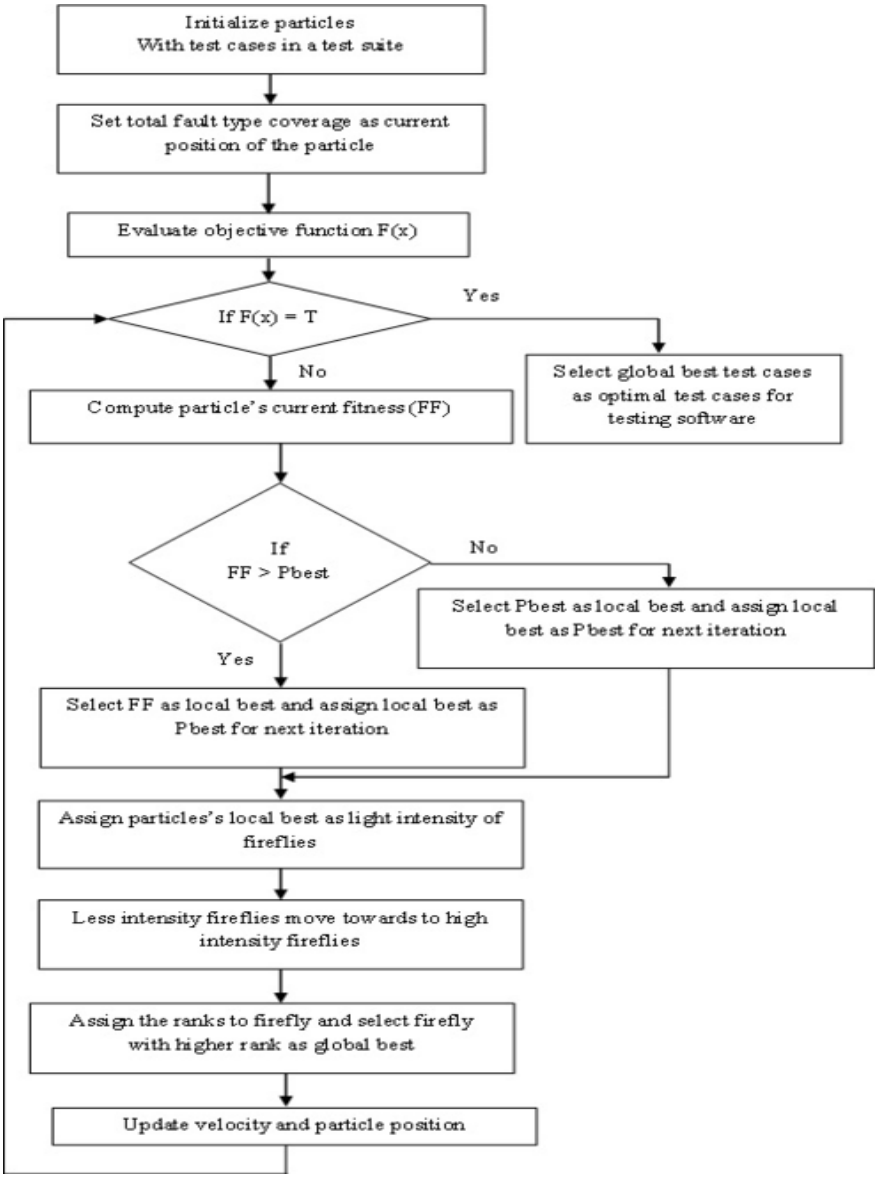
$$f\left(y\right)=\left\{f_1\left(y\right),\ f_2\left(y\right),\ f_3\left(y\right),\ldots\ f_k\left(y\right)\right\} \tag{1}$$

In (1), $f\left(y\right)$ denotes a set of 'k' criteria $f_1\left(y\right),f_2\left(y\right),f_3\left(y\right),\ldots f_k\left(y\right)$ where 'y' represents an individual solution for the problem being solved. The proposed HMCPW-RFMO techniquesolves the problem of optimal test case selection with multiple criteria by the hybridization of particle swarm and ranked firefly metaheuristic optimization technique. The HMCPW-RFMO technique uses five different criterions such as pairwise coverage, execution cost, fault detection capability, fault tolerance capability and average execution frequency. The different criteria are formulated as follows,

$$f\left(y\right)=\left\{f_1\left(y\right),\ f_2\left(y\right),\ f_3\left(y\right),f_4\left(y\right),f_5\left(y\right)\right\} \tag{2}$$

In (2), $f\left(y\right)$ denotes the set of 'k' criteria. $f_1\left(y\right)$ denotes a pairwise fault type coverage, $f_2\left(y\right)$ represents a execution cost, $f_3\left(y\right)$ denotes a fault detection capability, $f_4\left(y\right)$ is the fault tolerance capability, $f_5\left(y\right)$ is the average execution frequency. For each particle, the fitness is calculated with

Figure 2. Flow process of HMCPW-RFMO technique



multiple criteria. The pairwise coverage is the number of user requirements (i.e. feature pairs) are covered by a test case in a test suite. The pairwise fault type coverage capability is computed as follows,

$$PC = \frac{f_c}{t_s} \tag{3}$$

In equation (3), PC denotes a pairwise capability, f_c denotes the number of fault feature pairs (user requirements) covered by a test case, t_s represents a total number of feature pairs for testing the product lines in a given software program applications. The other criteria $f_2\left(y\right)$ is the execution cost which is defined as the amount of time taken for executing the test cases in a given test suite to test the software quality during combinatorial testing. Execution cost is computed as follows,

$$EC = \sum_{i=1}^{n} T\left(tc_i\right) \tag{4}$$

In (4), EC indicates an execution cost, T denotes an average time required for executing the number of test cases tc_i in test suite. $f_3\left(y\right)$ represents a fault detection capability aimed at finding the errors in software programs. Besides, fault detection capability refers to the ratio of successful execution of the test case in a specified time which is defined as follows,

$$FDC = \frac{\sum_{i=1}^{n} tc_i}{N} \tag{5}$$

In (5), FDC denotes a fault detection capability tc_i represents set of test a cases that are successfully executed in a particular time. N denotes the number of test cases. From that, a test case with higher FDC value is selected for combinatorial testing. The other objective function $f_4\left(y\right)$ denotes a fault tolerance capability it means that the system continues the operation in the presence of the bounded number of failures occurs. The final criterion is the average execution frequency $(f_5\left(y\right))$ which estimates the number of test cases executed recurrently in a particular period of time based on domain knowledge. The maximum average execution frequency value of a test case is selected as optimal for combinatorial testing. It is computed as follows,

$$AF = \frac{\sum_{i=1}^{n} ef_{tc_i}}{N} \tag{6}$$

In (6), AF refers the average execution frequency of test case in the particular period of time, N denotes a total number of test cases. Based on this above said criteria, fitness is computed as follows,

$$FF = \max f_1\left(y\right) + \min f_2\left(y\right) + \max f_3\left(y\right) + \max f_4\left(y\right) + \max f_5\left(y\right) \tag{7}$$

In (7), FF denotes a fitness function of each particles, $\max f_1\left(y\right)$ denotes a HMCPW-RFMO techniqueselects the test cases with maximum fault type coverage, minimum execution cost $\min f_2\left(y\right)$, maximum fault detection capability $\max f_3\left(y\right)$, maximum fault tolerance capability $\max f_4\left(y\right)$ and maximum average execution frequency $\max f_5\left(y\right)$. The minimum and maximum result is obtained by setting the certain threshold value. If the current computed value is greater than the threshold value, then the maximum results is obtained. If the current computed value is lesser than the threshold

value, then the minimum results is obtained.Based on this above criteria, hybrid optimization technique selects the optimal test cases for combinatorial testing. After calculating the fitness of the particles, the comparisons of the two particles takes place for finding the local best particles. Each current fitness of the particle is compared with fitness of previous best particles through the fitness value. If the current fitness value of the particles is better than the previous fitness of the particle(Pbest), then set the current fitness value of the particle as the new Pbest (i.e. local best). As a result, the local best test cases are selected based on general particle swarm optimization technique. In order to find the global best solution, the ranked firefly metaheuristic optimization is applied. This approach is designed based on the flashing light behaviour of fireflies. Here the local best test cases are represented as 'n' number of fireflies. The entire fireflies are unisexual where any fireflies get attracted by the other fireflies based on their light intensity (i.e. fitness function). The attractiveness of the firefly is related to the brightness. The firefly with less intensity is attracted to the brighter one. Then it moves in arbitrarily to other brighter one having higher fitness and the intensity of firefly gets increased. Whereas the intensity of the firefly is reduced having the lesser fitness of the current best solutions. From the results, the brighter one is selected as a global best solution. Let us consider the number of fireflies (local best test cases) $f_1, f_2, f_3 \ldots . f_n$. For each firefly, the light intensity is formulated with the fitness function FF .

$$I\left(f_i\right) = FF \tag{8}$$

In (8), $I\left(f_i\right)$ denotes a light intensity of the firefly. The firefly with high light intensity attracts the other one i.e. $I\left(f_j\right) > I\left(f_i\right)$ where $j = 1,2,3,\ldots n$ but $i \neq j$. The attractiveness of the firefly is changed along with the degree of light absorption. Based on the intensity measure, the less brightness firefly f_i moves towards its other brighter f_j . After that, the rank is assigned to the all the local best fireflies,

$$r \rightarrow \ f_1, f_2, f_3 \ldots . f_n \tag{9}$$

In (9), r denotes a rank which is assigned to all the fireflies based on their brightness level. The firefly with a higher rank is selected as a global best solution. The lesser ranked firefly gets removed. Then, in order to find the local best particle (Pbest) for next iteration, the position of the particle is updated as follows,

$$tc_{i+1} = tc_i + v_{i+1} \tag{10}$$

In (10), tc_{i+1} denotes an updated position of the particles. tc_i denotes a current position, v_{i+1} represents a updated velocity. The velocity of the particle is updated using (11).

$$v_{i+1} \quad = \mid \mathrm{N}_{\mathrm{fault}} \mid \tag{11}$$

In (11), $\mid \mathrm{N}_{\mathrm{fault}} \mid$denotes number of new faults covered by global test case. Whereas,

$$\mathrm{N}_{\mathrm{fault}} = \mathrm{G}_{\mathrm{fault}} - \mathrm{L}_{\mathrm{fault}} \tag{12}$$

Algorithm 1. Hybrid Multi-Criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization

Hybrid Multi-criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization Algorithm
Input: Software product lines, Set of test cases in test suite $T = \left\{ tc_1, tc_2, ..tc_n \right\}$
Output: Select optimal test cases for combinatorial testing
Begin
1. Initialize the particles (i.e. test cases) with random position
2. For each tc_i
3. Calculate the FF using (7)
4. ***if*** $\left(\boldsymbol{FF} \geq \boldsymbol{Pbest} \right)$ then
5. Assign current particle as local best
6. else
7. Keep previous one is a local best
8. end if
9. For each local best firefly tc_i
10. Formulate light intensity $I\left(f_i\right)$ based on fitness function FF
11. if $\left(\boldsymbol{I}\left(\boldsymbol{f_j}\right) > \boldsymbol{I}\left(\boldsymbol{f_i}\right) \right)$ then
12. Firefly f_i moves towards f_j
13. end if
14. Rank the f_i based on their light intensity
15. Select high rank f_i as global best
16. End for
17. Update v_{i+1} using (11)
18. Update tc_{i+1} using (10)
19. Go to step 2 while the maximum iteration is attained
20. End for
End

In (12), N_{fault} denotes set of new faults type covered by global test case. G_{fault} denotes set of faults type covered by global test cases and L_{fault} denotes set of faults type covered by local test cases. The above process is repeated until the maximum fault type coverage is attained. By this way, the test case reduction is performed. With the selected global optimum test case, the software product lines are tested to improve the software quality. The algorithmic process of the HMCPW-RFMO technique is described as follows,

Algorithm 1 describes the Hybrid Multi-Criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization to find the global best test cases for combinatorial software testing to improve the software quality. In HMCPW-RFMO, Initialize total number of test suites in software as swarm size and number of test suites as number of test cases for the hybrid optimization. Initially, total faults type covered by each particle is considered as current position of the particle and velocity of the particle as zero. For each test case, there are five different criteria is considered such as pairwise fault type coverage, execution cost, fault detection capability, fault tolerance capability and average execution of the frequency. Based on these criteria, the fitness function is computed. If the fitness of the particle is greater than the Pbest, then the particle is assigned as a local best. Otherwise, the previous one is assigned as the local best. After finding the local best solution, the global best is selected using

ranked firefly metaheuristic optimization technique. Here the firefly is considered the local best test cases. The light intensity of the firefly is formulated with the fitness function. If the light intensity of one firefly is lesser than the other one, then it travels to the other brighter one. Then the brightness of the firefly gets improved. Followed by, the rank is assigned to the entire Firefly based on their light intensity. The high ranked firefly is selected as a global optimum solution. Then the velocity and current position of the particles are updated for making the comparisons to select the local best. The above process is repeated until the maximum iteration is attained.

4.0 EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed HMCPW-RFMO technique and existing methods namely (Bestoun S. Ahmed, 2016) Cuckoo search algorithm (CSA) and (Annibale Panichella et al., 2015) Diversity based Genetic Algorithm (DIV-GA) are implemented in Java Language using schoolmate data set. The SchoolMate dataset comprises the number of open-source programs to perform software quality test. The open-source program contains the number of product lines. Software tester performs the combinatorial testing and monitors every product lines in the source program with the optimal test cases. The SchoolMate dataset is taken from https://sourceforge.net/projects/schoolmate/?source=directory. The SchoolMate dataset comprises numerous test suites like administration, teachers, students and parents etc,. From the schoolMate dataset, we considered a trial input fault matrix with 8 particles as rows and 10 faults type as column are presented in Table 1. Fault matrix can be encrypted using binary values either 0 or 1, where 1 denotes a fault type FT_i covered by its related particle P_i and 0 denotes that a fault type is not covered. Therefore, amount of faults type covered by related particle test case can be identified using trial input fault matrix. Table 2 shows step by step implementation of proposed HMCPW-RFMO technique with trial input fault matrix presented in Table 1. Proposed HMCPW-RFMO required only 4 particle test cases {p0, p6, p2, p4} to obtain all faults type as shown in Table 2 and also HMCPW-RFMO is implemented in Java language and execution outcomes are shown in Fig 3. From the results observed in Fig 3, we conclude that only 4 test cases and 0.017(ms) computational time are needed to cover all faults type which improves maximum test suite reduction rate.

The HMCPW-RFMO technique experimented with CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015). The experimental evaluation is conducted with the parameters such as testcase reduction rate, computation time, fault coverage rate and testing cost. The experimental results of the HMCPW-RFMO technique and existing methods are discussed in the following section.

5.0 RESULTS AND DISCUSSIONS

Outcomes of the HMCPW-RFMO technique is compared with existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) are described in this section. The performance analysis is carried out with a number of factors such as test case reduction rate, computation time, fault coverage rate and testing cost with respect to the number of test cases and product lines. The results are discussed in the following subsections using either table or graphical representation.

5.1. Impact of Test Case Reduction Rate

The Test case reduction rate is defined as the ratio of a number of optimal test cases are selected to the total number of test cases in a test suite. The test case reduction rate is mathematically formulated as,

$$TCRR = \frac{n - \ Number\ of\ test\ cases\ are\ selected\ as\ optimal}{n} * 100 \tag{14}$$

Table 1. Trial input fault matrix

Test case particle (P_i)	Faults type (FTi)										Total faults type covered	Faults type covered
	FT_1	FT_2	FT_3	FT_4	FT_5	FT_6	FT_7	FT_8	FT_9	FT_{10}		
P_1	0	1	0	1	0	0	1	0	1	0	4	2,4,7,9
P2	1	0	1	0	0	0	0	0	0	0	2	1,3
P3	1	0	0	0	1	0	1	1	0	0	4	1,5,7,8
P4	0	1	0	1	0	0	0	0	1	0	3	2,4,9
P5	0	0	1	0	0	1	0	0	0	1	3	3,6,10
P6	1	0	0	0	0	0	1	0	0	0	2	1,7
P7	0	0	1	0	0	1	0	1	0	0	3	3,6,8
P8	0	1	0	0	0	0	0	0	0	1	2	2,10

In (14), $TCRR$ denotes a test case reduction rate, n denotes a number of test cases. Test case reduction rate is measured interms of percentage (%).

Sample calculation for Test Case Reduction Rate

Ø **HMCPW-RFMO:** Total number of test cases and thenthe number of test cases selected as optimal for combinatorial testing, then the $TCRR$ is computed as follows,

$$TCRR = \frac{10-4}{10} * 100 = 60\%$$

Ø **Existing CSA:** Total number of test cases and the number of test cases selected as optimal for combinatorial testing, then the $TCRR$ is computed as follows,

$$TCRR = \frac{10-5}{10} * 100 = 50\%$$

Ø **Existing DIV-GA:** Total number of test cases and the number of test cases selected as optimal for combinatorial testing, then the $TCRR$ is computed as follows,

$$TCRR = \frac{10-6}{10} * 100 = 40\%$$

Fig. 4 depicts the performance results of test case reduction rate in the given test suite versus a number of test cases. For the experimental consideration, the numbers of test cases are varied from 10 to 100. While varying the number of test cases, the different reduction rate is obtained as shown in Fig 4. The above graphical results clearly show that the test case reduction rate is considerably improved using HMCPW-RFMO technique when compared to the other two existing methods CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015). This is because, the HMCPW-RFMO technique uses a hybrid optimization technique for selecting the optimal test cases among the number of test cases in the given test suite. The test suite comprises a number of test cases. Among the number of test cases, optimal test cases are selected for performing the combinatorial testing. This process decreases the test cases. The test case minimization is achieved by applying a hybrid multi-criteria particle swarm and ranked firefly metaheuristic optimization. This hybrid technique

Table 2. Step by step implementation of proposed HMCPW-RFMO technique with trial input fault matrix presented in Table 1

Iteration	Particle	Current position (TC_i)	Faults type covered (L_{fault})	Fitness (FF)	Old Pbest	Local best (P_{best})	Gbest firefilies with higher rank (G_{fault})	N_{fault} (G_{fault} - L_{fault})	Velocity v_{i+1} = | N_{fault} |	Updated Particle position (TC_{i+1})
1	P_0	4	2,4,7,9	0.4	0.0	0.4 *	P_0 {2,4,7,9}	-	0	4
	P_1	2	1,3	0.2	0.0	0.2		2,4,7,9	4	6
	P_2	4	1,5,7,8	0.4	0.0	0.4		2,4,9	3	7
	P_3	3	2,4,9	0.3	0.0	0.3		7	1	4
	P_4	3	3,6,10	0.3	0.0	0.3		2,4,7,9	4	7
	P_5	2	1,7	0.2	0.0	0.2		2,4,9	3	5
	P_6	3	3,6,8	0.3	0.0	0.3		2,4,7,9	4	7
	P_7	2	2,10	0.2	0.0	0.2		4,7,9	3	5
2	P_0	4	2,4,7,9	0.4	0.4	0.4	P_6 {2,3,4,6,7,8,9}	3,6,8	3	7
	P_1	6	1,2,3,4,7,9	0.6	0.2	0.6		6,8	2	8
	P_2	7	1,2,4,5,7,8,9	0.7	0.4	0.7		3,6	2	9
	P_3	4	2,4,7,9	0.4	0.3	0.4		3,6,8	3	7
	P_4	7	2,3,4,6,7,9,10	0.7	0.3	0.7		8	1	8
	P_5	5	1,2,4,7,9	0.5	0.2	0.5		3,6,8	3	8
	P_6	7	2,3,4,6,7,8,9	0.7	0.3	0.7 *		-	0	7
	P_7	5	2,4,7,9,10	0.5	0.2	0.5		3,6,8	3	8
3	P_0	7	2,3,4,6,7,8,9	0.7	0.4	0.7	P_2 {1,2,3,4,5,6,7,8,9}	1,5	2	9
	P_1	8	1,2,3,4,6,7,8,9	0.8	0.6	0.8		5	1	9
	P_2	9	1,2,3,4,5,6,7,8,9	0.9	0.7	0.9 *		-	0	9
	P_3	7	2,3,4,6,7,8,9	0.7	0.4	0.7		1,5	2	9
	P_4	8	2,3,4,6,7,8,9,10	0.8	0.7	0.8		1,5	2	10
	P_5	8	1,2,3,4,6,7,8,9	0.8	0.5	0.8		5	1	9
	P_6	7	2,3,4,6,7,8,9	0.7	0.7 *	0.7		1,5	2	9
	P_7	8	2,3,4,6,7,8,9,10	0.8	0.5	0.8		1	1	9
4 *	P_0	9	1,2,3,4,5,6,7,8,9	0.9	0.7	0.9	P_4 {1,2,3,4,5,6,7,8,9,10}	10	1	10
	P_1	9	1,2,3,4,5,6,7,8,9	0.9	0.8	0.9		10	1	10
	P_2	9	1,2,3,4,5,6,7,8,9	0.9	0.9 *	0.9		10	1	10
	P_3	9	1,2,3,4,5,6,7,8,9	0.9	0.7	0.9		10	1	10
	P_4 *	10 *	1,2,3,4,5,6,7,8,9,10	1 *	0.8	1		-	0	10
	P_5	9	1,2,3,4,5,6,7,8,9	0.9	0.8	0.9		10	1	10
	P_6	9	1,2,3,4,5,6,7,8,9	0.9	0.7	0.9		10	1	10
	P_7	9	1,2,3,4,6,7,8,9,10	0.9	0.8	0.9		5	1	10

initializes the number of particles and their positions. Then the fitness of the test cases is measured with multi-criteria. Based on these criteria, the local best test cases are selected. Among the local best test cases, the global best test cases are selected by applying ranked firefly optimization. The light intensity of the firefly is computed based on the fitness function. If the light intensity of one firefly is higher than the other one, the lower brightness is moved towards a higher one. As a result,

Figure 3. Execution results of HMCPW-RFMO technique

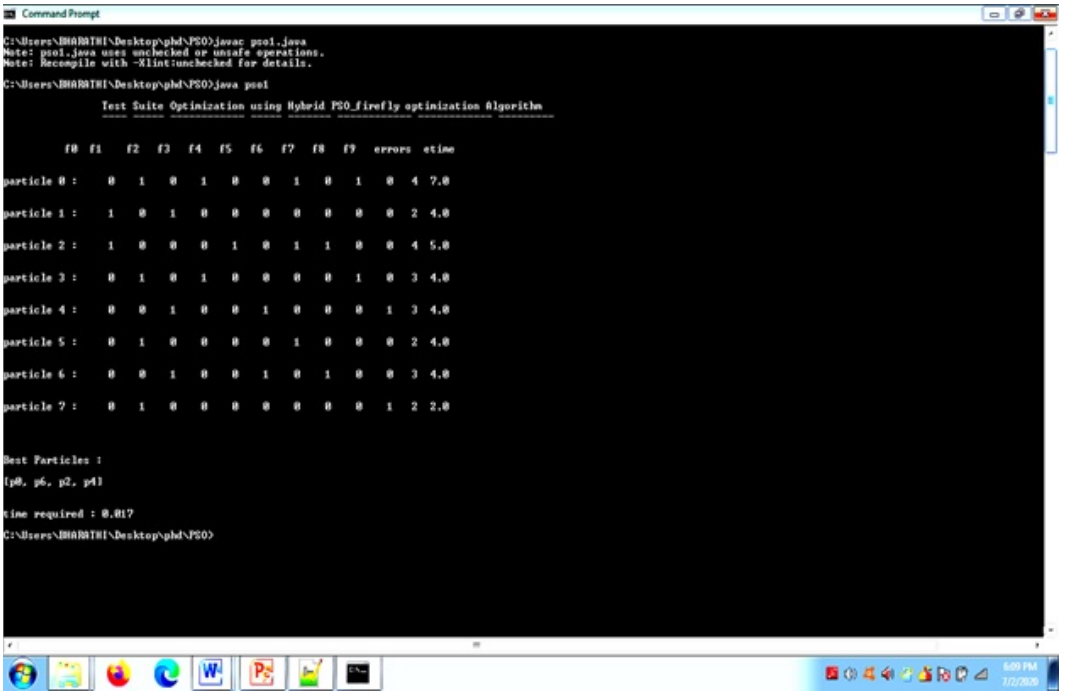
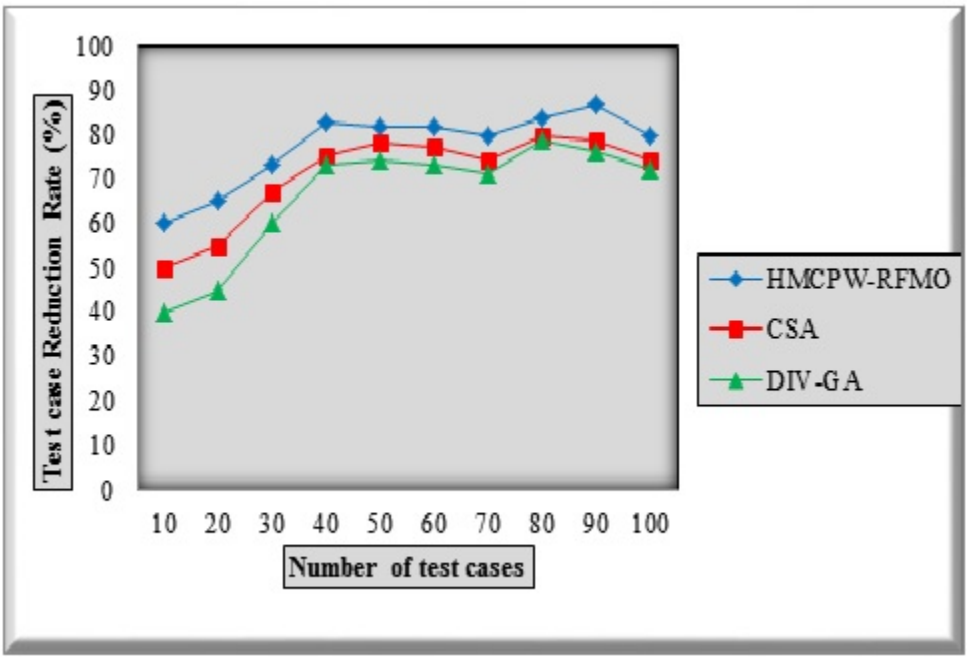


Figure 4. Performance results of test case reduction rate



the brightness of the firefly gets increased. firefly getsrank is assigned to the entire Firefly based on their brightness level. The firefly with high rank is selected as optimal. Compared to existing methods, the HMCPW-RFMO technique selects less number of optimal test cases for testing the software program resulting in improving the software quality. Totally ten runs are carried out with a number of test cases. The average result of test case reduction rate using HMCPW-RFMO technique is considerably increased by 10% and 20% when compared to existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) respectively.

5.2. Impact of Computational Time

The Computational Time is defined as an amount of time taken for selecting the optimal test cases for combinatorial testing. The computational time is measured as follows,

$$ct = Number\ of\ test\ cases^{*}T\left(selecting\ one\ test\ case\right) \tag{15}$$

In (15), ct denotes acomputation time, T denotes an amount of time taken for selecting the optimal test cases in the given test suites. The computation time is measured in terms of milliseconds(ms).

Sample calculation for computation time

Ø **HMCPW-RFMO: N**umber of test cases 10 and the time for detecting the optimal test cases is 0.9ms, then the computation time is computed as follows,

$$ct = 10^{*}0.9ms = 9ms$$

Ø **Existing CSA: N**umber of test cases 10 and the time for detecting the optimal test cases is 1.5ms, then the computation time is computed as follows,

$$ct = 10^{*}1.5ms = 15ms$$

Table 3. Tabulation for computation time

Number of test cases	Computation time (ms)		
	HMCPW-RFMO	CSA	DIV-GA
10	9	15	19
20	14	18	24
30	18	24	30
40	23	27	32
50	25	30	36
60	31	37	41
70	36	41	48
80	42	45	49
90	44	47	52
100	49	53	60

Ø **Existing DIV-GA: N**umber of test cases 10 and the time for detecting the optimal test cases is 1.9ms, then the computation time is computed as follows,

$ct = 10{*}1.9ms = 19ms$

Table 3 describes the performance results of computation time with the number of test cases. Totally ten runs are carried out with a different number of test cases varied from 10 to 100. The above computation time performance result is considerably improved using HMCPW-RFMO technique when compared to existing methods. The HMCPW-RFMO technique takes minimum time to select the optimal test cases for testing the software program which results in improving the software quality. This significant improvement of the proposed HMCPW-RFMO technique performs multi-criteria optimizations such as pairwise coverage, execution cost, a fault detection capability, fault tolerance capability and average execution frequency. Based on these multi-criteria, the HMCPW-RFMO technique selects the optimal test cases with minimum time.

The above table values show that the HMCPW-RFMO technique takes 9ms for optimal test case selection while considering the 10 test cases. Whereas the other existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) technique uses 15ms and 19ms for selecting the optimal test cases to perform combinatorial testing. Followed by, the remaining nine runs are performed with a number of test cases. The above discussion shows that the HMCPW-RFMO technique minimizes the time taken for test case selection in combinatorial testing. The average comparison result shows that the HMCPW-RFMO technique minimizes the computational time by 17%when compared to existing CSA (Bestoun S. Ahmed, 2016). In addition, the computation time of HMCPW-RFMO technique is minimized by 29% when compared to existing DIV-GA (Annibale Panichella et al., 2015)technique.

5.3. Impact of Fault Coverage Rate

Fault Coverage rate is defined as the ratio of the number of optimal test cases that covers the more faults in the given product lines. Coverage rate is computed using the following mathematical formula,

$$FCR = \frac{n - no.of\ \ optimal\ \ testcases\ \ thatcovers\ \ morefaults}{n} {*}100 \tag{16}$$

In (16) FCR denotes a fault coverage rate and 'n' denotes the number of test cases. Fault coverage rate is measured in terms of percentage (%).

Sample calculation for Coverage Rate

Ø **HMCPW-RFMO: N**umber of test cases is 10 and the number of optimal test cases covers more faults are 3, and then the fault coverage rate is calculated as follows,

$$FCR = \frac{10-3}{10} {*}100 = 70\%$$

Ø **Existing CSA: N**umber of test cases is and the number of optimal test cases covers more faults are 5, and then the fault coverage rate is calculated as follows,

$$FCR = \frac{10-5}{10} {*}100 = 60\%$$

Ø **Existing DIV-GA: N**umber of test cases is 10and the number of optimal test cases covers more faults are 4, and then the fault coverage rate is calculated as follows,

$$FCR = \frac{10-5}{10} * 100 = 50\%$$

Figure 5. Performance results of fault coverage rate

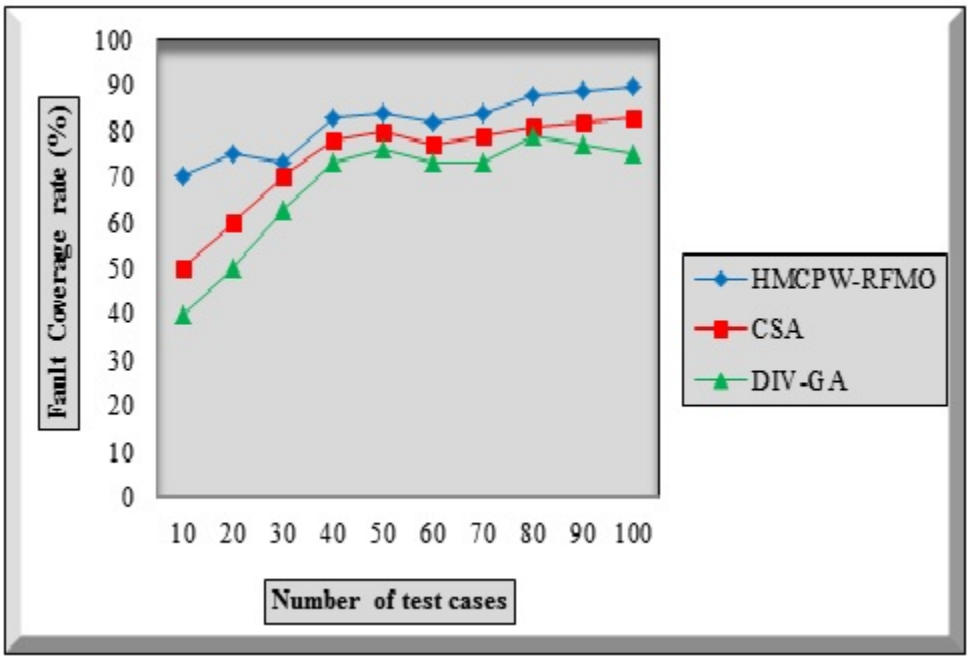


Fig. 5 illustrates the fault coverage rate versus a number of test cases. The number of test cases taken for experimental evaluation is varied from 10 to 100. In Fig. 5, three different color lines indicate the fault coverage rate of three different methods namely HMCPW-RFMO technique and existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015). Among the performance of three different methods, the HMCPW-RFMO technique improves the fault coverage rate using less number of test cases. The hybrid optimization technique selects the local best and global best test cases. The multi-criteria particle swarm optimization selects the local best test cases among the number of test cases. The ranked firefly metaheuristic approach combined with the particle swarm optimization to find the global best test cases. Among the selected test cases, the test case which covers the number of faults in the software program is identified. Followed by, the quality of the software program gets improved. In case of existing methods, a number of optimal test cases are selected for fault coverage resulting in improving the complexity. On the contrary, the HMCPW-RFMO technique selects less number of optimal test cases to perform fault coverage. This helps to minimize the complexity and improves the software quality.

After performing the ten runs, the comparison between proposed and existing methods are carried out. The average comparison results show that the fault coverage rate of proposed HMCPW-RFMO is significantly improved by 12% and 24% when compared to existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) respectively.

5.4. Impact of Testing Cost

Testing Cost defined as the amount of time required for testing the software with the optimized test cases in the test suite. The testing cost is calculated mathematically as follows,

$$T_c = E_t - S_t \tag{17}$$

In (17), T_c denotes a testing cost, E_t denotes an ending time of the software testing, S_t denotes a starting time of the software testing. Testing cost is measured in terms of milliseconds (ms).

Sample calculation for testing cost

Ø **HMCPW-RFMO:** Let us consider the size of software product lines is 15KB, software testing ending time is 9ms and software testing starting time is 0ms, then the testing cost is computed as follows,

$$T_c = 9ms - 0ms = 9ms$$

Ø **Existing CSA** Let us consider the size of software product lines is 15KB, software testing ending time is 13ms and software testing starting time is 0ms, then the testing cost is computed as follows,

$$T_c = 13ms - 0ms = 13ms$$

Ø **Existing DIV-GA:** Let us consider the size of software product lines is 15KB, software testing ending time is 18ms and software testing starting time is 0ms, then the testing cost is computed as follows,

$$T_c = 18ms - 0ms = 18ms$$

Table 4 describes the performance results of testing cost with respect to the size of the software product lines. For the experimental consideration, ten different sizes of the software product lines are taken from 15KB to 150KB. For the different sizes of software product lines, the testing cost is measured. The different testing cost is obtained for the different size of the software program. As shown in the Table 4, the proposed HMCPW-RFMO technique minimizes the testing time while performing the combinatorial testing. The proposed hybrid optimization technique performs combinatorial testing with the optimal number of test cases. This optimal test case maximizes the coverage and reduces the testing cost of the software program. In other words, HMCPW-RFMO technique selects less number of test cases for testing the software product lines. This, in turn, minimizes the testing cost. Let us consider the 15KB size of software product lines, the testing cost of HMCPW-RFMO technique is 9ms. The testing cost of other existing methods CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) are 13ms and 18ms respectively. Followed by, the remaining nine runs are carried out and compared the existing methods. The average comparison results of testing cost using HMCPW-RFMO technique is minimized by 19% and 34% when compared to existing CSA (Bestoun S. Ahmed, 2016) and DIV-GA (Annibale Panichella et al., 2015) respectively.

The above results and discussions clearly show that the HMCPW-RFMO technique considerably improves the software quality by improving the test case reduction rate, fault coverage rate with minimum testing cost as well as computation time.

Table 4. Tabulation for testing cost

Size of Software Product Lines (KB)	Testing cost (ms)		
	HMCPW-RFMO	CSA	DIV-GA
15	9	13	18
30	11	15	20
45	15	21	25
60	17	20	26
75	18	24	29
90	24	30	35
105	26	29	37
120	30	33	36
135	31	36	43
150	37	41	48

6.0 CONCLUSION

An efficient technique called, Hybrid Multi-Criteria Particle Swarm and Ranked Firefly Metaheuristic Optimization (HMCPW-RFMO) is developed for improving the test case reduction with minimum testing cost. The HMCPW-RFMO technique takes a number of test cases as input for performing the optimization process. The Multi-Criteria Particle Swarm optimization method selects the local and global best test cases among the number of test cases. Initially,the fitness of each test case is calculated based on the Multi-Criteria. After that, the fitness is compared with the pbest. If the fitness of the test cases is better than the pbest, then the test case is assigned as local best. Then the global best is identified by using Ranked Firefly Metaheuristic approach. The locally selected test cases are ranked based on their brightness level. The high ranked test cases as a global best solution of the hybrid optimization technique. Then the combinatorial software testing is performed with these selected optimal test cases. As a result, the software quality gets improved. Finally, the quality of the proposed and existing methods is analysed with the schoolmate datasets with different parameters such as test case reduction rate, computation time, fault coverage rate and testing cost. The performance results show that the HMCPW-RFMO technique improves test case reduction rate, fault coverage rate with minimum computation time as well as testing cost when compared to existing methods.

REFERENCES

Ahmed, Sahib, & Potrus. (2014). Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing. *Engineering Science and Technology, an International Journal*, *17*, 218-226.

Ahmed, B. S., Abdulsamad, T. S., & Potrus, M. Y. (2015). Achievement of Minimized Combinatorial Test Suite for Configuration-Aware Software Functional Testing Using the Cuckoo Search Algorithm. *Information and Software Technology*, *66*, 13–29. doi:10.1016/j.infsof.2015.05.005

Ahmed. (2016). Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, *19*, 737–753.

Avila-George, H., Torres-Jimenez, J., Gonzalez-Hernandez, L., & Hernández, V. (2013). Metaheuristic approach for constructing functional test-suites. *IET Software*, *7*(2), 104–117. doi:10.1049/iet-sen.2012.0074

Haider, Ali, Nadeem, Aamer, Rafiq, & Shahzad. (2014). Multiple objective test suite optimization: A fuzzy logic based approach. *Journal of Intelligent & Fuzzy Systems*, *27*(2), 863-875.

Karsoliya, , Sinhal, & Kanungo. (2013). Combined Architecture for Early Test Case Generation and Test suit Reduction. *International Journal of Computer Science Issues*, *10*(1), 484–489.

Mahmoud, T., & Ahmed, B. S. (2015). An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use. *Expert Systems with Applications*, *42*(22), 8753–8765. doi:10.1016/j.eswa.2015.07.029

Nasser, A. B., Zamli, K. Z., Alsewari, A. R. A., & Ahmed, B. S. (2018). Hybrid flower pollination algorithm strategies for t-way test suite generation. *PLoS One*, *13*(5), 1–24. doi:10.1371/journal.pone.0195187 PMID:29718918

Panichella, A., Oliveto, R., Penta, M. D., & De Lucia, A. (2015). Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms. *IEEE Transactions on Software Engineering*, *41*(4), 358–383. doi:10.1109/TSE.2014.2364175

Petke, J., Cohen, M. B., Harman, M., & Yoo, S. (2015). Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection. *IEEE Transactions on Software Engineering*, *41*(9), 901–924. doi:10.1109/TSE.2015.2421279

Qi, X.-F., Wang, Z.-Y., Mao, J.-Q., & Wang, P. (2017). Automated Testing of Web Applications Using Combinatorial Strategies. *Journal of Computer Science and Technology*, *32*(1), 199–210. doi:10.1007/s11390-017-1699-x

Soares de Souza, L. (2015). A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection. *Journal of the Brazilian Computer Society*, *21*(19), 19. doi:10.1186/s13173-015-0038-8

Soma, S. B. L. (2012). Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony. *Procedia Engineering*, *30*, 191–200. doi:10.1016/j.proeng.2012.01.851

Wang, Y., Zhou, M., Song, X., Gu, M., & Sun, J. (2018). Constructing Cost-Aware Functional Test-Suites Using Nested Differential Evolution Algorithm. *IEEE Transactions on Evolutionary Computation*, *22*(3), 334–346. doi:10.1109/TEVC.2017.2747638

Xing, Y. (2015). A Hybrid Intelligent Search Algorithm for Automatic Test Data Generation. *Mathematical Problems in Engineering*, 1–15.

Yilmaz, C. (2013). Test Case-Aware Combinatorial Interaction Testing. *IEEE Transactions on Software Engineering*, *39*(5), 684–706. doi:10.1109/TSE.2012.65

Yilmaz, C., Dumlu, E., Cohen, M. B., & Porter, A. (2014). Reducing Masking Effects in Combinatorial Interaction Testing: A Feedback Driven Adaptive Approach. *IEEE Transactions on Software Engineering*, *40*(1), 43–66. doi:10.1109/TSE.2013.53

Zamli, K. Z., Din, F., Kendall, G., & Ahmed, B. S. (2017). An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation. *Information Sciences*, *399*, 121–153. doi:10.1016/j.ins.2017.03.007

Zhang, Z., Yan, J., Zhao, Y., & Zhang, J. (2014). Generating combinatorial test suite using combinatorial optimization. *Journal of Systems and Software*, *98*, 191–207. doi:10.1016/j.jss.2014.09.001

Zheng, W., Wu, X., Hu, D., & Zhu, Q. (2016). Xiaoxue Wu., Desheng Hu., & Qihai Zhu.(2016). Locating Minimal Fault Interaction in Combinatorial Testing. *Advances in Software Engineering*, *2016*, 1–10. doi:10.1155/2016/2409521

M. Bharathi is working as an Assistant professor in the Department of Computer Science, Government Arts and Science College, Pennagaram, Dharmapuri, Tamilnadu, India. She has completed her MCA during the year of 2005 and M.Phil (Computer Science) during the year 2009 and completed her Ph.D in the year of 2020. She has been specialized in this area of software engineering, Data mining, Compiler design, and programming languages, Python and data science. She has attended many national and international conferences and presented several papers. She has fifteen years of experience in the field of computer science. Delivered Guest Lecture at various Engineering and Arts Colleges. She has supervised several research scholars and her aim to produce more number of valuable research works to this environment.