IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

Bio-Inspired Gradient Genetic Optimization for Test Suite Generation

T. Ramasundaram[1*], V.Sangeetha[2]

[1]Department of Computer Science, Periyar University, Salem, Tamil Nadu, India
[2]Department of Computer Science, Periyar University Constituent College, Pappireddipatti, Tamil Nadu, India

[*]*Corresponding Author: profram01@yahoo.in*

Available online at: www.ijcseonline.org

Accepted: 13/Jan/2019, Published: 31/Jan/2019

***Abstract*—** Software testing is an essential process during the software development process. Test suite generation process is employed to detect test cases with sources. Recently, many research works have been developed for automatically generate the software test suites. However, software testing is a time consuming and unable to obtain high coverage rate. In this paper, Gradient Advanced Genetic Parameter Control Based Test Suite Generation (GAGPC-TSG) technique is proposed. Based on the fitness value, the best test case is selected using roulette wheel selection. Later, the gradient approach is applied to obtain the optimal test case to generate the test suites for increasing the software quality. This enhances the better performance in terms of optimal test suite generation with minimum time and maximum fault coverage rate.

***Keywords*—** Software testing, test cases, roulette wheel selection, gradient approach

I. INTRODUCTION

Testing is the most significant parts of the software development process but is normally manual, error-prone and expensive. Software testing is the process of creating reliable, robust, and trustworthy software by executing a system to detect failures. However, it is still time- consuming process. In some testing circumstances, many faults are detected over a period of time. In software testing, automatically generating a test suite with high coverage plays a significant concern to software engineers for improving the software quality. But it is delayed by the optimal tests suites generation. A collection of test cases are often called a test suite for testing the software programs with some specified behaviors. In general, a software developer has less knowledge about the test suites generation. Therefore the controlling of some parameters is the demanding issues to obtain high coverage rate and improve the software quality. The several research works have been developed for software test suite generation. In this work, the search based technique is used to generate high coverage test suites.

A test case minimization approach was introduced for reducing the test cases in configuration-aware structural software testing [1]. The approach uses a cuckoo search (CS) along with a combinatorial approach for generating the test suites with optimal test cases. The approach consumes higher computational time for creating the optimal test suites.

Artificial Bee Colony (ABC) Optimization Based on Markov Approach was developed to achieve software code coverage with the optimal test suites [2]. The coverage rate of the ABC optimization was not improved effectively to improve software quality.

A regeneration genetic algorithm (RGA) was introduced for generating the software test data with high coverage [3]. The algorithm does not obtain optimal test suites with minimum time. A High-Level Hyper-Heuristic Tabu search strategy was introduced for t-way test suite generation [4]. The strategy does not improve the test suite generation with the optimal test cases.

An improved Ant Colony Optimization technique was developed for creating the optimal test cases and also improving the coverage capability [5]. The technique does not select the optimal test cases with minimum time. Archive-based Whole Test Suite Generation was performed in [6]. The approach failed to cover more faults during the software testing.

A new and effective approach called fuzzy-based adaptive swarm optimization algorithm was designed for software-testing activities [7]. The performance of the algorithm does not improve significantly by designing an effective data structure to enable the strategy for the specific combination. Complete Controllable test suites were generated for distributed software testing based on the size of pairwise differentiates test suites in [8]. The generated test suite does not satisfy user requirements.

An intelligent water drop (IWD) optimization algorithm was introduced for improving the entire software code coverage

with the number of test cases [9]. The algorithm does not optimize the test case using some other meta-heuristic approach. An efficient Cuckoo search Algorithm (CA) was designed for optimizing the test data with minimum time [10]. The algorithm does not apply to large datasets with various constraints.

The several issues are identified from the above said issues are less fault coverage capability, failure to improve software quality, more computational time for test suite generation and so on. Such kind of problems is overcome by introducing a novel technique called Gradient Advanced Genetic Parameter Control Based Test Suite Generation (GAGPC-TSG) technique is introduced.

The major contribution of the proposed GAGPC-TSG technique is described as follows:

1. First, the GAGPC-TSG technique generates the population of test cases for testing the given software program. The fitness is verified for each test case to generate the optimal test suites. Based on the fitness value, optimal test cases are selected among the populations.
2. The fitness criterion is not satisfied, then the GAGPC-TSG technique uses roulette wheel selection to choose the current best test cases among the population based on the fitness value using wheel pointer. Then the two-point crossover is applied to swap the two strings and creating the two offspring. Then the bit flip mutation changes the one bit randomly in the offspring.
3. The gradient approach is applied to control the crossover and mutation probability value to get an optimal test case for generating the test suite using adjustment coefficients. This helps to improve the high fault coverage rate and improve the precision of test suite generation with minimum time.

The rest of the paper is organized as follows. The works related to our objective is discussed in Section 2. Section 3 provides a detail explanation of the proposed GAGPC-TSG technique with neat diagram. Section 4 provides the experimental evaluation with the dataset. Results and discussion of the proposed and existing methods are described in section 5 for showing the performance of the proposed method. Finally, the conclusion of the research work is presented in section 6.

II. RELATED WORK

A new model was introduced for generating the whole test suites with the aim of increasing the covering rate in [11]. The model does not generate the optimal test suite with minimum time. A hybrid intelligent algorithm was developed for obtaining efficient software test data [12]. The algorithm failed to reach the high coverage capability in software testing.

A Simulated Annealing (SA) and Greedy Algorithms were developed to find the combinatorial interaction testing (CIT) test suites [13]. The algorithm has less fault coverage capability. An ant colony algorithm was designed for automatically generating graphical user interface (GUI) test cases in [14]. The algorithm does not have the high fault detection capability with the help of generated GUI test cases. A new model based test design architecture (MBTDA) was introduced for test suite creation with minimum time and cost [15]. The model failed to improve the accuracy of the optimal test suite generation.

A Neuro-fuzzy modeling-based approach was developed to optimize the regression test suite and also provides better performance with less execution time of the test suite generation in [16]. The method failed to generate optimal test cases for improving the coverage rate. Artificial Bee Colony (ABC) based search method was presented for generating the software test suite and achieves the entire test coverage [17]. The optimization technique consumes more time for generating the test suites.

Two multi-objective optimization algorithms namely Swarm Optimization and harmony search algorithm were developed for multi-objective test case selection [18]. The optimization algorithms failed to use the number of programs for software testing with the selected test cases. A Hierarchical clustering approach was designed for minimizing the test suite and increasing the coverage rate in [19]. The approach takes more time for test suite generation. A Fuzzy clustering approach was developed for optimizing the software test suite and also obtaining high coverage [20]. The fuzzy clustering approach does not reduce the software test suite generation time.

Correlation feature subset algorithm was developed to increase software quality using micro interaction metrics [21]. But, the developed algorithm was unable to generate optimal test suites. Adaptive neuro-fuzzy inference system was introduced for identifying software faults and reducing the cost of software implementation [22]. However, the time taken for predicting software defects was failed to reduce. Open source software technology was developed to increase the quality and reliability of the software system [23]. But, fault coverage ratio remained unaddressed.

The issues identified by the above-said techniques are higher test suite generation time, less fault detection capability and coverage rate and so on. In order to overcome the issues, the Gradient Advanced Genetic Parameter Control Based Test Suite Generation (GAGPC-TSG) technique is developed and it explained in the following sections.

III. METHODOLOGY

Software testing is the process of executing a system to detect failures and improve software quality. Software test

suite optimization is the significant concern in software testing with the optimal test cases. During test suite generation, a different optimization method has been developed to unit test generation. Specifically, the search-based unit test suite generation is obtained based on the number of test cases. In software testing, a test suite is a validation suite to test software programs. The optimal test suites are generated with the number of test cases. Then testing is performed with these test suites to improve the software quality. An efficient technique called Gradient Advanced Genetic Parameter Control Based Test Suite Generation (GAGPC-TSG) technique is developed. The architecture diagram of the GAGPC-TSG technique is shown in figure 1.

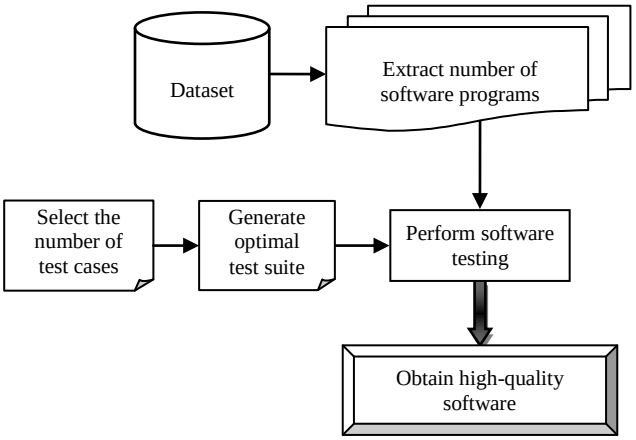

Figure1: Architecture Diagram of GAGPC-TSG Technique

Figure 1 shows the architecture diagram of the GAGPC-TSG technique to generate the best test suite for testing the software programs with the optimal test cases. During the test suite generation, the genetic parameter value is controlled by applying a search based technique. Initially, the software programs are extracted from the dataset for testing purposes. Then the test cases are selected for generating the test cases for testing the software programs with optimal test cases. In general, the selected test cases are used for satisfying the user requirements like total test duration, total test effort cost, Number of modules tested, number of faults detection and so on. Based on these requirements, the test suites are generated using gradient advanced genetic parameter control.

A gene optimization is a bio-inspired meta-heuristic search technique. It is often developed to resolve search and optimization problems. It is generally more feasible to evaluate the entire input space and it is used to generate good solutions in reasonable time by evaluating the input space. The first step in the functioning of a proposed GAGPC-TSG technique is an initial population generation. A population of candidate solutions also termed as an individual (i.e. the number of test cases) is initialized. Let us consider the software program SC, taken from the dataset and the population of test cases are randomly initialized for generating the test suites are expressed as follows,

$$\{tc_1, tc_2, tc_3 ....., tc_n\} \qquad (1)$$

From (1), T denotes a set of test cases $\{tc_1, tc_2, tc_3 ....., tc_n\}$. Among the several test cases, the optimal test cases are selected through the optimization to generate the best test suite for testing the software program. In GAGPC-TSG technique, the fitness of each individual in the population is calculated. The fitness function is an objective function which provides the optimal solution. It also produces the output solution to attain high coverage rate. It means the optimal value of the parameter is chosen depends on the user test requirements. The fitness of each individual (i.e. test case) is calculated as follows,

$$FF = \arg\max\{UR\} \qquad (2)$$

From (2), FF denotes a fitness function, arg max at which the function outputs (i.e. user required test cases) are as large as possible. It means the selected test case maximize the user test requirements. Based on the fitness calculation, the optimal test cases are selected and generate the unit test suit and it satisfies the user requirements. Therefore, proposed GAGPC-TSG technique generates test suites with higher coverage capability for testing the software program. If the optimal value of the parameters is not obtained, then the genetic operators such as selection, crossover, and mutation are performed.

A. Roulette wheel test case selection

Roulette wheel test case selection is used for choosing the best individual based on the fitness value. Let us consider a circular wheel for selecting the best individuals from the population. The circular wheel is partitioned into 'n' number of segments, where 'n' is the number of test cases in the population. In this selection, all the test cases in the population are positioned on the roulette wheel based on their fitness value. The selection of the best individual is shown in figure 2.

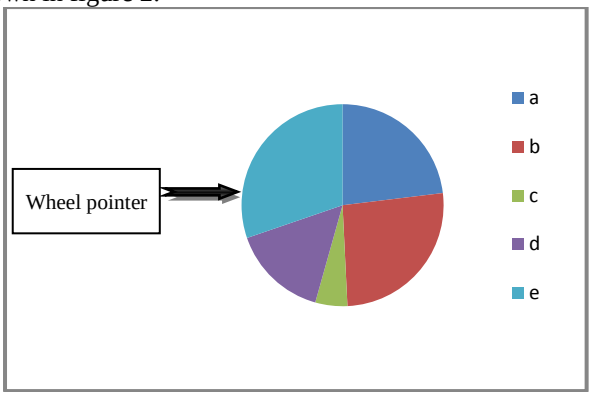

Figure 2: Roulette Wheel Based Test Case Selection

Figure 2 shows the best individual selection using a roulette wheel with the help of the wheel pointer. From the figure, the

different segment color indicates the fitness value of the different individuals a, b, c, d, and e. The roulette wheel is rotated. The individual of the wheel which comes in front of the wheel pointer is selected and it has the high fitness function. As a result, a test case with high fitness has a high chance for selection. The probability of selecting the best individuals from the population is expressed as follows,

$$P = \frac{FFi}{\sum_{j=1}^{n} FF_j} \quad (3)$$

From (3), P denotes a selection probability, FF_j denotes a fitness of individual 'i' in the population 'j'. 'n' is the number of individuals in the population. Based on the above probability, individuals with high fitness are selected for recombination.

B. Two-point crossover for offspring generation

Chromosome encoding is the representation of an individual. In the test suite generation, the bit string encoding methods are used for the recombination process. Let us consider the bit string representation of the individual is denoted as '1' and '0'. The proposed technique uses two-point crossovers for generating the offspring. Let us consider two parents a = 1111101010 and b = 0100100111. Then the offspring is generated as sown in figure 3.

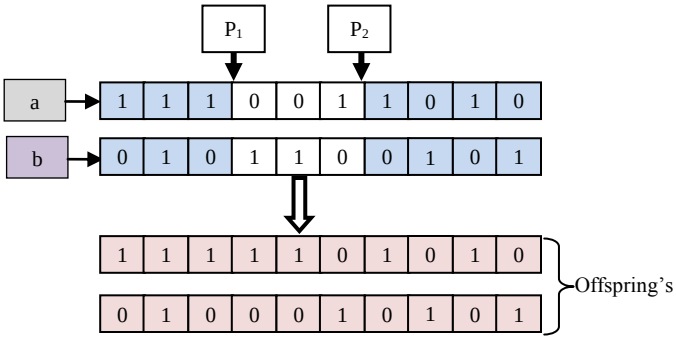

Figure 3: Two-point Crossover Based Offspring Generation

Figure 3 shows the process of two-point crossover based offspring generation. The two offspring are generated with the two cross point P_1 and P_2. Based on the crossover results, the two individuals are recombined with their string values to get new off-springs. After creating the new offspring's, the string length of offspring is similar to the total string length of both the parents. Crossover is the significant process which helps to generate the test suites using a number of test cases. The value of the two-point crossover probability P_{tc} is controlled by introducing the gradient approach. The probability of the crossover is defined as follows,

$$P_{tc} = [g_l(s)]^2 + [h_l(s)]^2 \quad (4)$$

From (4), P_{tc} denotes a two-point crossover probability $g_l(s)$ is the similar strings in the length of the two offspring are generated and $h_l(s)$ denotes a dissimilar string in the length of the two offspring are generated. Based on the results, mutation probability is measured. As a result of crossover, the process is not able to create diversity within a population. Therefore, diversity is preserved by using mutation operators.

C. Bit flips Mutation

Once the new offspring are generated, the other genetic operator called mutation is performed. Mutation is a process of random variation in the given string. It is also used to preserve genetic diversity from one generation of a population to the next generation. The proposed GAGPC-TSG technique uses bit-flip mutation is used for randomly interchanging the bit. This mutation operator takes the newly generated offspring's from the two-point crossover and inverts the bits '1' to '0' and vice versa at a random position. The bit flip mutation is shown in figure 4.

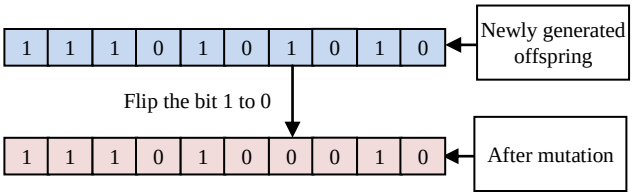

Figure 4: Bit Flip Mutation

Figure 4 shows the process of bit flip mutation to obtain a new solution. The above figure shows that the bit '1' is randomly changed into a bit '0'. The probability of the mutation rate is denoted as P_{BM}. The bit flip mutation probability is calculated as follows,

$$PBM = \frac{Na}{l * P_{tc} * 100} \quad (5)$$

From (5), N_a denotes a population size in integer l denotes a length of the string in integer, P_{tc} represents a two-point crossover probability. Finally, the crossover and mutation rates are controlled using the gradient approach. The Gradient approach is a repair operator to provide feasible solutions in problems with equality constraints (i.e. the probability value of the crossover and mutation is equal to 1).

$$\beta \rightarrow P_{BM} + P_{tc} = 1 \quad (6)$$

From (6), β denotes a sum function of both crossover and mutation probability. The adjustment of genetic parameters using the gradient operator is described as follows,

$$\Delta = \begin{Bmatrix} \beta + \theta; if \beta < 1 \\ \beta - \theta; if \beta > 1 \end{Bmatrix} \quad (7)$$

From (7), Δ denotes a gradient operator θ denotes an adjustment coefficient of the control parameters. By applying the gradient approach, the genetic parameters crossover and mutation are controlled. After that, the old individual is replaced into a new one. Then the fitness is calculated for a new individual (i.e. test case) based on the user requirements. Then the Fitness criterion is verified to obtain optimal solutions.

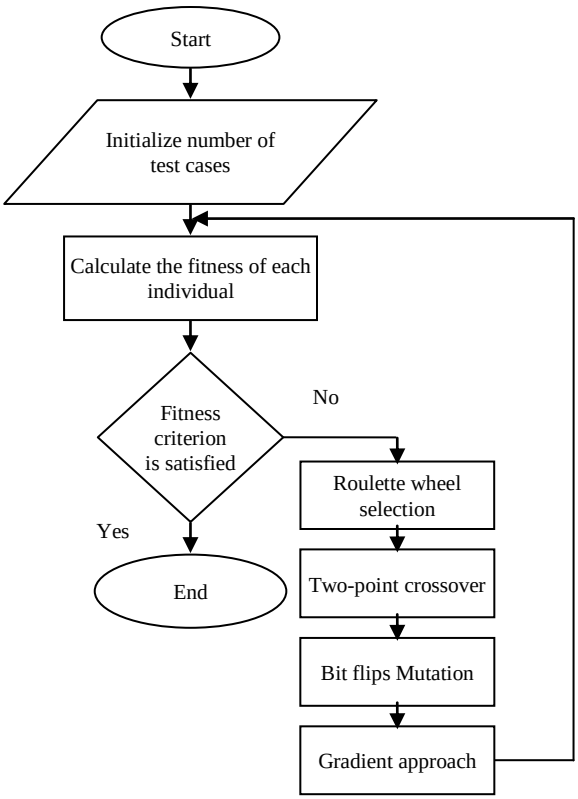

Figure 5: Flow Process of GAGPC-TSG Technique

This process is continued for all possible test cases in an initial population and generates a new test suite. The flow process of the proposed GAGPC-TSG technique is described as shown in figure 5.

Figure 5 shows the flow process of a proposed GAGPC-TSG technique to generate a test suite with the optimal test cases for testing the given software programs based on user test requirements. Initialize the number of test cases to generate the optimal test suite. For each individual, the fitness criterion is verified with the user requirements. As a result, efficient test cases are identified and these collections of the test cases are used to generate the test suite. If the fitness criterion is not satisfied, then the selection, crossover, and mutation are carried out to find the optimal test cases.

Then the crossover and mutation value are controlled by applying a gradient approach. This process generates the best test suites for software quality testing. This also helps to improve the coverage rate of faults in a software program. The algorithmic description of the proposed GAGPC-TSG technique is described as follows,

Input: Number of cases $\{tc_1, tc_2, tc_3 ....., tc_n\}$
Output: Generate test suite with optimal test cases
Begin
1. Initialize number of test cases tc_n
2. **For** each tc
3. calculate the fitness FF
4. **if** *the fitness criterion* is satisfied then
5. select optimal test case
6. **else**
7. Select the individuals from the population based on probability P
8. Swap two individuals to generate offspring's
9. Invert the input bit using mutation probability
10. Adjust P_{BM} and P_{tc} using a gradient operator Δ
11. Replace old individual
12. Go to step 3
13. **end if**
14. **end for**
15. Obtain optimal test suites
end

Algorithm 1.Gradient Advanced Genetic Parameter Control Based Test Suite Generation

The algorithmic process of proposed GAGPC-TSG technique is described to generate the test suite for covering the number of faults with the generated optimal test cases. In order to test the software quality, the program is taken from the dataset. The population of a number of test cases is randomly initialized. Then the fitness of each test case is calculated with the user requirements. If the selected test cases are not satisfied the user requirements, the roulette wheel selection is applied to choose the individual from the population based on the fitness probability. Based on the selected individual, the recombination is carried out to create the new offspring's. Then the random bit flip mutation is performed to interchange the bit and is used for obtaining the

global optimal solution. After mutating the bits, the gradient approach is applied to control the crossover and mutation probability rate based on the adjustment coefficient. Then the fitness function is measured and selects the user required test cases. As a result, the test suites are generated with the optimal test cases.

IV. EXPERIMENTAL SETTINGS

Experimental evaluation of the proposed Gradient Advanced Genetic Parameter Control Based Test Suite Generation (GAGPC-TSG) technique is implemented using JAVA programming with Webchess dataset http://sourceforge.net/projects/webchess/. The Webchess dataset comprises 28 PHP Programs for software testing with the generated test suites. The input program is taken from this dataset. Then the generated test suites used for testing the input program. Each and every line in the source program is monitored to cover the more faults and increase the software quality.

Experimental evaluation is performed with three different methods namely GAGPC-TSG technique, Test case minimization approach [1] and Artificial Bee Colony Optimization (ABC) [2]. The different parametric values are obtained and the discussions are explained in the following sections.

V. RESULTS AND DISCUSSION

Results and discussion of the proposed GAGPC-TSG technique are performed with the existing methods namely Test case minimization approach [1] and Artificial Bee Colony Optimization (ABC) [2] with certain parameters such as precision, test suite generation time and coverage rate with the given input program. The performance is evaluated according to the following metrics with the help of table and graph values.

A. Impact of precision

Precision is defined as the ratio of the number of optimal test suites generated from the total test suites generated. The precision is measured as follows,

$$\Pr ecision = \frac{no.\ of\ optimal\ test\ suites\ generated}{n} * 100 \quad (8)$$

From the equation (8), where 'n' denotes the number of the test suite. The precision is measured in terms of percentage (%).

Sample calculation for precision

GAGPC-TSG: no. of optimal test suites generated is 8 and the number of the test suite is 10, then,

$$\Pr ecision = \frac{8}{10} * 100 = 80\%$$

Test case minimization approach: no. of optimal test suites generated is 7 and the number of the test suite is 10, then the

$$\Pr ecision = \frac{7}{10} * 100 = 70\%$$

ABC optimization technique: no. of optimal test suites generated is 6 and the number of the test suite is 10, then the precision is measured as,

$$\Pr ecision = \frac{6}{10} * 100 = 60\%$$

Table 1: Tabulation for Precision

No. of test suites	Precision (%)		
	GAGPC-TSG	Test case minimization approach	ABC optimization technique
10	80	70	60
20	85	75	65
30	83	73	67
40	80	70	60
50	90	80	72
60	93	85	75
70	89	79	71
80	95	89	84
90	94	88	81
100	96	90	82

Table 1 describes the performance results of precision with the number of test suites. The precision is the accuracy of optimal test suites generation. The above table clearly shows that the performance result of precision is significantly improved using GAGPC-TSG technique when compared to existing methods. This improvement is achieved by applying gradient advanced gene parameter control based high coverage test suite generation. By applying advanced gene parameter control method, the number of test cases is selected from the population. Initially, the population of test cases is randomly initialized. Then the test cases are verified with the fitness condition. It means the test case satisfies the user requirements are selected to generate the optimal test suite. If the condition is not satisfied, the operators such as selection, crossover, and mutation are carried out. The selection is performed using the circular wheel. The circular wheel selects the test cases with the high fitness. The wheel pointer indicates the current best individual from the population. Then the two-point crossover is applied to interchange the chromosome values. As a result of interchanging, the new offspring's are generated from the parent chromosome. After that, the mutation probability is calculated. The bit flip mutation is used to randomly changing the bit from the newly generated offspring's. Then the probability of the mutation is computed with the crossover probability rate. Finally, the gradient approach is

applied to control the crossover and mutation probability rate. Then they obtained test cases are verified with the fitness condition and select the test suites. This process is repeated for each test case in the population. Finally, the selected test cases are combined to generate the number of optimal and high coverage test suites. This optimal test suite is more suitable for software testing. Totally ten runs are carried out to show the results of proposed and existing methods. The results reveal that the GAGPC-TSG technique considerably improves the precision by 11% and 24% than the existing methods Test case minimization approach [1] ABC optimization technique [2].

B. Impact of test suite generation time

Test suite generation time is the amount of time required for generating the optimal test suites. Test suite generation time is calculated using the following mathematical equations,

$$TSGT = n * T(optimal\ test\ suite\ generation) \quad (9)$$

From (9), TSGT denotes a test suite generation time, 'n' denotes the number of test suites, T represents a time. Test suite generation time is measured in terms of milliseconds (ms).

Sample calculation for test suite generation time:

GAGPC-TSG: no. of optimal test suites generated is 10 and the time for one optimal test suite generation is 0.1ms, then

$$TSGT = 10 * 1 = 10ms$$

Test case minimization approach: no. of optimal test suites generated is 10 and the time for one optimal test suite generation is 1.6ms, then the

$$TSGT = 10 * 1.6 = 16ms$$

ABC optimization technique: no. of optimal test suites generated is 10 and the time for one optimal test suite generation is 1.9ms, then the

$$TSGT = 10 * 1.9 = 19ms$$

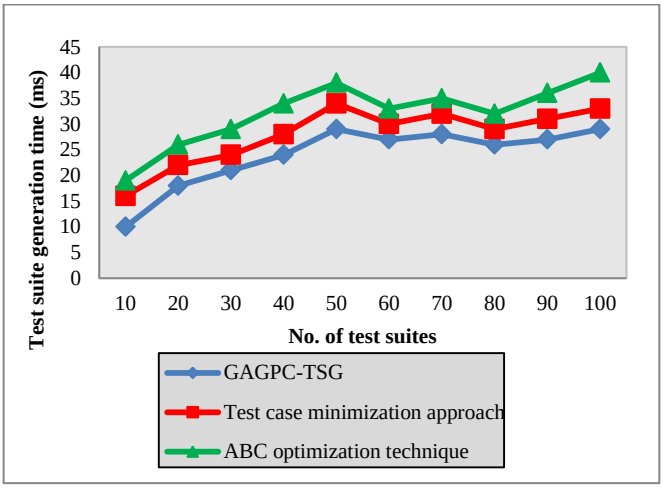

Figure 6: Performance Results of Test Suite Generation Time

Figure 6 shows the performance results of test suite generation time versus a number of test suites. The numbers of test suites are varied from 10 to 100. While varying the number of test suites, the generation time gets varies as shown in figure 6. The above graphical representation shows that the test suite generation time is considerably minimized using proposed GAGPC-TSG when compared to existing methods. This significant improvement is obtained using gradient advanced gene parameter control. This optimization technique takes minimum time for test suites generation with the number of optimal test cases. The process of selecting the test cases which helps to find faults in the software programs based on the user requirements. Let us consider, the software program is tested with the generated test suites. This test suite satisfies the user requirements like a number of faults are correctly detected, minimum testing duration and so on. The gradient advanced gene parameter control technique calculates the fitness for selecting the test cases among the population. If the fitness criterion is not satisfied, the roulette wheel selection is applied to choose the current best fitness test cases. Then the crossover is used to swap the two individuals to provide the offspring. Then the offspring's are mutated to obtain the diversity. The gradient approach controls the crossover and mutation probability rate. These processes are used to generate the optimal test suites with minimum time. As a result, the user required test cases are selected and generated the optimal test suites. The input program is tested with this optimal test suite for improving the software quality. The optimal test suite generation time is significantly reduced by 16% and 27% when compared to the existing Test case minimization approach [1] ABC optimization technique [2] respectively.

C. Impact of the coverage rate

Coverage rate is defined as the ratio of the number of test suites covers the faults in the given program. Coverage rate is measured as follows,

$$coverage\ rate = \frac{no\ of\ optimal\ test\ suites\ that\ covers\ more\ faults\ in\ program}{n} * 100 \quad (10)$$

From (10), where 'n' denotes the number of generated test suites. Coverage rate is measured in terms of percentage (%).

Sample calculation for coverage rate:

GAGPC-TSG: no. of optimal test suites that covers more faults in a given time is 7 and the total number of test suite generated is 10, then

$$coverage\ rate = \frac{7}{10} * 100 = 70\%$$

Test case minimization approach: no. of optimal test suites that covers more faults in a given time is 6 and the total number of test suite generated is 10, then

$$coverage\ rate = \frac{6}{10} * 100 = 60\% .$$

ABC optimization technique: no. of optimal test suites that covers more faults in a given time is 5 and the total number of test suite generated is 10, then

$$\text{cov}\,erage\ rate = \frac{5}{10} * 100 = 50\%$$

Table 2: Tabulation for Coverage Rate

No. of test suites	Coverage rate (%)		
	GAGPC-TSG	Test case minimization approach	ABC optimization technique
10	70	60	50
20	75	65	55
30	77	67	57
40	78	68	55
50	82	72	60
60	87	77	68
70	84	73	66
80	90	81	75
90	88	79	74
100	92	80	77

Table 2 describes the coverage rate versus a number of test suites. The number of test suites taken for experimental evaluation is varied from 10 to 100. Table 2 shows the coverage rate of three methods namely GAGPC-TSG technique and existing Test case minimization approach [1] Artificial Bee Colony Optimization (ABC) technique [2]. The above table value clearly shows that the coverage rate of the proposed GAGPC-TSG technique is considerably improved than the other existing methods. This is because the GAGPC-TSG technique uses the gradient approach during the optimization. The gradient approach effectively generates the test suites with optimal test cases. The numbers of test cases are taken as population for the optimization. The gradient approach controls the two gene parameters with their probability value. The probability value of the mutation and crossover rate is controlled by using a gradient operator. If probability value is either lesser or greater one, then the adjustment coefficient controls the operators. After that, the fitness of the test cases is calculated with their user requirements. Similarly, the test cases are selected from the populations. The selected test cases are combined to generate the test suite which is the optimal one. This helps to cover the entire defects in the software program. The comparison results show that the coverage rate of proposed GAGPC-TSG technique is considerably improved by 14% and 30% when compared to existing Test case minimization approach [1] ABC optimization technique [2] respectively.

VI. CONCLUSION

In this paper the proposed GAGPC-TSG technique for improving software quality. The analysis and experimental results show that efficient GAGPC-TSG technique has the maximum performance in generating optimal test suites than test case minimization approach and artificial bee colony optimization. Here, Webchess dataset has been used in which 10 to 100 test suites are considered for evaluating the GAGPC-TSG technique. The proposed GAGPC-TSG technique improves the precision by 18% and fault coverage rate by 22% and reduces the test suite generation time by 22%.

REFERENCES

[1] Bestoun S. Ahmed, "*Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing*", Engineering Science and Technology, an International Journal, Elsevier, Vol.**19**, pp. 737–753, **2016.**

[2] Muthusamy Boopathi, Ramalingam Sujatha, Chandran Senthil Kumar, Srinivasan Narasimman, "*Quantification of Software Code Coverage Using Artificial Bee Colony Optimization Based on Markov Approach*", Arabian Journal for Science and Engineering, Springer, Vol.**42,** Issue. **8,** pp. 3503–3519, **2017.**

[3] Shunkun Yang, Tianlong Man, Jiaqi Xu, Fuping Zeng, Ke Li, "*RGA: A lightweight and effective regeneration genetic algorithm for coverage-oriented software test data generation*", Information and Software Technology, Elsevier, Vol.**76,** pp. 19–30, **2016.**

[4] Kamal Z. Zamlia, Basem Y. Alkazemib, Graham Kendall, "*A Tabu Search hyper-heuristic strategy for t-way test suite generation*", Applied Soft Computing, Elsevier, Vol.**44**, pp. 57–74, **2016**.

[5] Shunkun Yang, Tianlong Man, and Jiaqi Xu, "*Improved Ant Algorithms for Software Testing Cases Generation*", The Scientific World Journal, Hindawi Publishing Corporation, Vol.**2014**, May pp. 1-9, **2014**.

[6] José Miguel Rojas, Mattia Vivanti, Andrea Arcuri, Gordon Fraser, "*A detailed investigation of the effectiveness of whole test suite generation*", Empirical Software Engineering, Springer, Vol.**22,** Issue. **2**, pp. 852–893, **2017**.

[7] Thair Mahmoud 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, Elsevier, Vol.**42**, Issue. **22**, pp. 8753-8765, **2015**.

[8] Robert M. Hierons, "*Generating Complete Controllable Test Suites for Distributed Testing*", IEEE Transactions on Software Engineering, Vol.**41**, Issue. **3**, pp. 279 – 293, **2015**.

[9] Komal Agarwal, Manish Goyal, and Praveen Ranjan Srivastava, "*Code coverage using intelligent water drop (IWD)*", International Journal of Bio-Inspired Computation, Vol.**4**, Issue. **6**, pp. 392-402, **2012**.

[10] Manju Khari and Prabhat Kumar, "*An Effective Meta-Heuristic Cuckoo Search Algorithm for Test Suite Optimization*", Vol.**41**, Issue. **3**, pp. 363–377, **2017**.

[11] G. Fraser and A. Arcuri, "*Whole Test Suite Generation*", IEEE Transactions on Software Engineering, Vol.**39**, Issue. **2**, pp. 276 – 291, **2013**.

[12] Ying Xing, Yun-Zhan Gong, Ya-Wen Wang, and Xu-Zhou Zhang, "*A Hybrid Intelligent Search Algorithm for Automatic Test Data Generation*", Mathematical Problems in Engineering, Hindawi Publishing Corporation, Vol.**2015**, pp. 1-15, **2014**.

[13] Justyna Petke, Myra B. Cohen, Mark Harman, Shin Yoo, "*Practical Combinatorial Interaction Testing: Empirical Findings on Efficiency and Early Fault Detection*", IEEE Transactions on Software Engineering, Vol.**41**, Issue. **9**, pp. 901 – 924, **2015**.

[14] Y. Huang and L. Lu, "*Apply ant colony to event-flow model for graphical user interface test case generation*", IET Software, Vol.**6**, Issue: **1**, pp. 50 – 60, **2012**.

[15] Saurabh Karsoliya, Prof.Amit Sinhal, Er.Amit Kanungo, "*Combined Architecture for Early Test Case Generation and Test suit Reduction*", International Journal of Computer Science Issues, Vol. **10**, Issue. **1**, pp. 484-489, **2013**.

[16] Zeeshan Anwar, Ali Ahsan, and Cagatay Catal, "*Neuro-Fuzzy Modeling for Multi-Objective Test Suite Optimization*", Journal of Intelligent Systems, Vol.**25**, Issue. **2**, pp.1-24, **2015**.

[17] Soma Sekhara Babu Lam, M L Hari Prasad Raju, Uday Kiran M, Swaraj Ch Praveen Ranjan Srivastav, "*Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony*", Procedia Engineering, Elsevier, Vol.**30**, pp. 191-200, **2012**.

[18] Luciano Soares de Souza, Ricardo Bastos Cavalcante Prudencio and Flavia A. de Barros, "*A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection*", Journal of the Brazilian Computer Society, Vol.**21**, Issue. **19**, pp. 1-20, **2015**.

[19] Fayaz Ahmad Khan, Anil Kumar Gupta, Dibya Jyoti Bora, "*An Efficient Technique to Test Suite Minimization using Hierarchical Clustering Approach*", International Journal of Emerging Science and Engineering, Vol.**3** Issue. **11**, pp. 1-10, **2015**.

[20] Gaurav Kumar, Pradeep Kumar Bhatia, "*Software testing optimization through test suite reduction using fuzzy clustering*", CSI Transactions on ICT, Springer, Vol.**1**, Issue. **3**, pp. 253–260, **2013**.

[21] A. Sreepradha, "Measuring Software Quality Using Micro Interaction Metrics", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Vol.2, Issue. 5, pp.1-4, 2017.

[22] G. Rajendra, Dr. M. Babu Reddy, "Application of Adaptive Neural Fuzzy Inference System for the Prediction of Software Defects", International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Vol.2, Issue. 3, pp.1-5, 2017.

[23] Kumari Seema Rani, "Open Source Software: A Prominent Requirement of Information Technology", International Journal of Scientific Research in Network Security and Communication, Vol.6, Issue. 2, pp.1-6, 2018.

