Received: August 2, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Optimal Software Test Case Mechanism using Grey Wolf-FireFly Method

Parag Rastogi [1]*

[1]*Subharti Institute of Technology and Engineering, Swami Vivekanand Subharti University, Computer Science & Engineering, India*
* Corresponding author's Email: parag0305@gmail.com

Abstract: Software testing is a major technique for designing the software without fault and use the resources efficiently for the development of any software. The software bugs can be detected by the process of executing an application or a program. Test Case Generation (TCG) is a method to identify the test data and satisfy the software testing criteria. An automatic TC technique automatically determines where the TC or test data generates by utilizing search based optimization method. In this paper, Gref-Wolf and Firefly algorithm (GWFF) method is used for optimization of TC and generation of path coverage within the minimal execution time. In general, the challenging task is constraint handling in which the fitness function is directly affected by updating the positions of the searching agents. In proposed method, there is no direct relation between the fitness function and search agents, because the agents update their positions with respect to the locations of alpha, beta and delta. Experimental results showed that the GWFF method helped to generate path convergence within minimal execution time. Comparing to the existing methods such as Bee Colony Algorithm (BCA), Particle Swarm Optimization (PSO) and Cuckoo Search (CS), the proposed GWFF method provided better performance in terms of fitness values.

Keywords: Execution time, Firefly optimization algorithm, Gref-Wolf algorithm, Software testing, Test case generation.

1. Introduction

Software has become an indispensable part of day to day activities and is very significant in technological as well as economic development [1]. Nowadays, software is significantly used in many fields such as home appliances, nuclear-power-plants, automobiles, telecommunications, medical devices and so on [2]. A software testing process is a significant task and indispensable stage to build high quality software to make it free from bugs and defects and to improve the quality. The software quality estimation uses several factors such as reliability, efficiency, software functionality, testability and so on [3, 4]. Earlier researches focused on to reduce complexities and the failure rate of the system. It is a difficult task to compute the best cost in a large area with a population at the random movement of many components [5]. At present, the software testing takes more time and cost and makes the software development process as an expensive task. But, the cost of testing decreases with the reduction of testing time [6]. However, most of the software delivered without enough testing, which is due to marketing pressures and the aim to save testing time and cost, but delivering the software without sufficient testing may lead to loss of revenue [7].

The software testing is an essential technique and it is very helpful for software developers. Several existing research works implemented to improve the quality of the software and their improvement is noted. The PSO algorithm generates the path coverage data, then the search direction of next iteration depends on the previous iteration coverage data [8]. The fuzzy clustering method is utilized to decrease the testing period as well as the number of TCs. This methodology uses Cyclomatic complexity to define the full-fledged conditional coverage, but this technique was more expensive [9]. The combination of fuzzy logic and CS algorithm used for software cost prediction and it provided the accurate prediction rate [10]. An environmental factor of developing software classifies the two classes such as testing environment and operational environment. [11]. In this paper, GWFF method is applied for optimizing the TC and generation of path coverage. The traditional engineering issues has been prevalent among analysts and improved in different investigations. The heuristic strategies that have been received to optimize this issue are: GWO, PSO, BCA, CS and Ant Colony algorithm. However, the existing strategies gives most extreme cost though the proposed GWO able to discover a plan with the minimum cost. In general, the GWO indicates high performance in solving challenging issues, because of the operators that are intended to enable GWO to avoid local optima effectively and converge towards the optimum rapidly. Moreover, this paper studied ATM machine withdrawal operation related TC generated from the combinational system diagram graph are merged with the State Chart and Sequence Diagram Graph (SCSEDG). The TCs are optimized through proposed GWFF and within the minimal execution time maximizes the generation of path coverage.

The organization of the paper is as follows. The existing recent research works on TC optimization described in section II. The proposed system, generation and optimization of a TC by employing GWFF method described in section III. Section IV shows comparative experimental result. The section V, explains the withdrawal operation of ATM machine case study. Finally, the conclusion is made in section VI.

2. Literature Review

Several techniques for TC optimization suggested by various researchers are listed in this section. The below section contains existing methodologies used for TC, the advantages and the limitations faced by the method are briefly discussed.

R.K. Sahoo, D.P. Mohapatra, and M.R. Patra [12] implemented a Firefly search technique used for generating and optimizing the random TC with test data. The fitness function was used to select the values of test data for effective and efficient method. The algorithm was compared with existing methods and the results showed that the FireFly algorithm produced optimal results which were more accurate with less time. The automated generation of test data work efficiently to generate best firefly solution only for smaller programs.

M. Khari, P. Kumar, D. Burgos, and R.G. Crespo [13] proposed the improvement of an automated tool with significant components. The two essential components of software testing are test suite optimization and generation. This method provided a set of minimal test cases with maximum path coverage's compared to other algorithms. The automated fault detection was implemented by generating optimal test suite. The automated testing model should be hybridized for better results in every aspect and the algorithm required a large number of inputs.

M. Boopathi, R. Sujatha, C.S. Kumar, and S. Narasimman [14] proposed a hybrid technique namely Markov chain and Artificial Bee Colony (ABC) optimization methods those were used to achieve the software code coverage. A number of paths were generated using Linear-Code-Sequence-And-Jump (LCSAJ) coverage. The LCSAJ was employed to decrease the number of independent paths related to the paths generated by original path testing. The calculation of test tolerability and reliability of different kinds of critical software was difficult to calculate through ABC optimization with mutation testing.

B.S. Ahmed [15] developed a method for reducing the number of TCs in configuration-aware structural testing. The generation of optimized test suite was carried out by the combinatorial optimization algorithm for sampling the input configuration. The evaluation results showed that the use of CS in the combinatorial test suites generated better results for optimization. The strategy proved its effectiveness in detecting faults at programs by using the functional testing approach. The method consumes more time for detecting faults in programming languages.

R.K. Sahoo, S.K., Nanda, D.P. Mohapatra, and M.R. Patra, [16] proposed a hybrid BCA for generating and optimizing the test cases from combinational UML diagrams. The objective of this proposed Hybrid BCA was to optimize the test cases and generation of path coverage within the minimal execution time. This gave better results in comparison with particle swarm and Bee colony algorithm. The proposed system took less time to choose the best test path and it is more capable, reliable for the development of software. The method was unable to enhance the test case or test data generation for large programs. The approach consumed large amount of time.

To overcome the above addressed limitations, a GWFF method is implemented to produce an automatic and optimized test case with less execution time.

3. Proposed methodology

Model driven testing is a method that signifies the behavioural model and encodes the system performance with particular terms and conditions. The model includes a group of objects that express by variables and object relationship. This research work, obtained an automated optimized TC or test data with potential test paths from combinational system graphs. The production and improvement of TCs by GWFF optimization technique from combinational IML diagrams. Here, ATM machine based cash withdrawal operation is considered for generating the TCs using SCSEDG. By using GWFF, the TCs are optimized. The objective of the proposed method is to optimize the TCs and generation of path coverage with minimal execution time. The proposed method takes minimum time to select the best path by avoiding the local optima and it is more reliable for the development of software. While intergating the firefly algorithm with GWO, the proposed method will handle large program to find the optimal solutions.

3.1 Conversion of State Chart Diagram to State Chart Diagram Graph

State chart diagram is under UML that describe the time taken by a software system. It majorly consists of transitions of states. The state-chart diagram represents the different states and events and different effects change the state. Fig. 1 represents the state-chart diagram and a state chart diagram graph for the withdrawal task of an ATM. Table 1 represents the dependency table for overall operation of ATM which is shown in the state chart diagram graph.

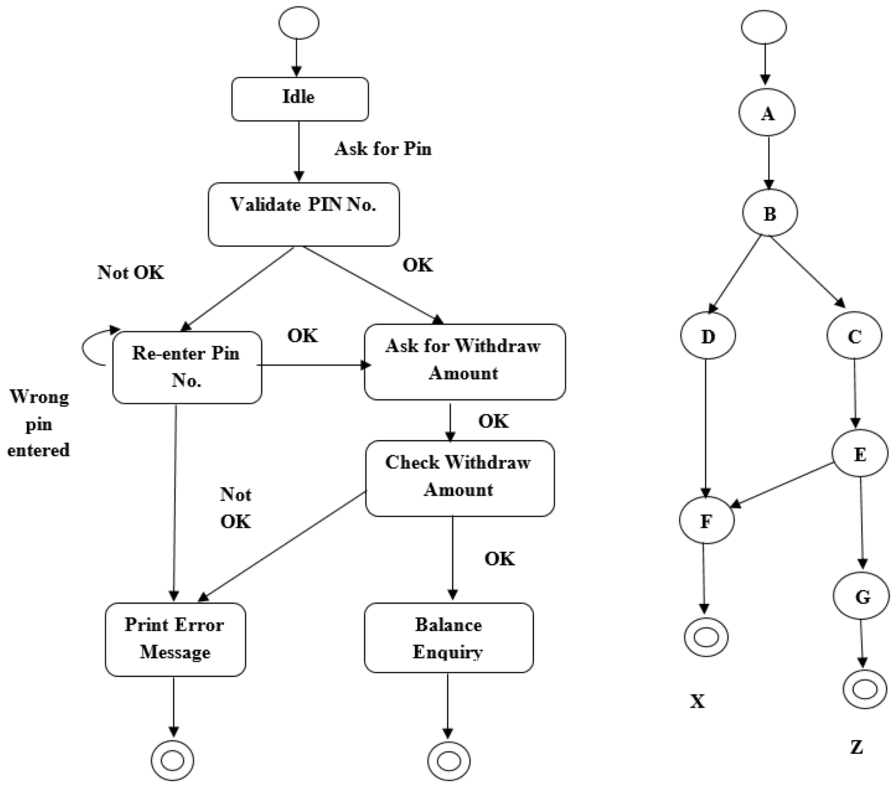

Figure.1 State chart diagram and state-chart diagram graph of overall operation of an ATM

Table 1. Dependency table of overall operation of an ATM through state-chart diagram graph

Symbol	Activity Name	Possible Number of Outputs	Dependency	Input	Expected Outputs
A	Insert Pin Number	1(B)	X	User promotes to enter Pin number	B: Pin number is forwarded for validation
B	Pin Number Validation	2(C,D)	A	Pin number provided by the user	C: Valid pin number
					D: Invalid pin number

D	Invalid Pin	1(Y)	B	Incorrect Pin number Message	F: Message displayed for incorrect pin number
C	Valid Pin	1(E)	B	Correct Pin entered message	E: Amount is forwarded for Checking
E	Amount Withdrawal	2(F, G)	C	User promotes to enter withdrawal amount	F: Withdrawal request not granted
					G: Request granted for valid withdrawal amount
F	Displaying Error Message	1(Y)	D	Invalid amount entered message	Y: Error message displayed
G	Balance Enquiry	1(Z)	E	Remaining balance after withdrawal operation	Z: Remaining Balance Message printed

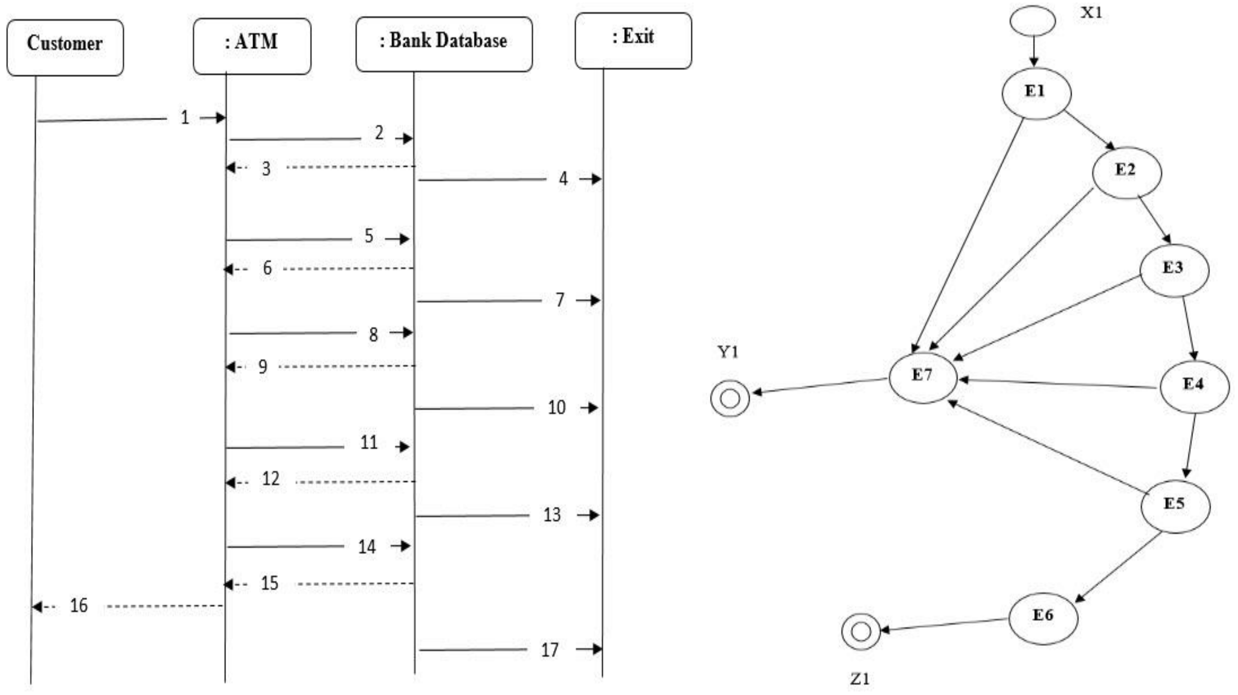

Figure.2 SCSEDG of withdrawal operation of an ATM

Where,

1. Check the withdrawal amount,

2. Check if negative or Zero.

3., 6., 9.,12., 15., Verified.

4., 7., 10., 13., Invalid amount.

Check withdrawal limit.

8. Check if multiple of 100.

11. Check today’s withdrawal limit

14. Check bank balance avaliability.

16. Valid amount message forwarded.

17. Incorrect amount message forwarded.

Table 2. Sequence diagram graph based dependency table of ATM machine withdrawal operation

Symbol	**Activity Name**	**Possible Number of Outputs**	**Dependency**	**Input**	**Expected Outputs**
E1	Check if amount is non-negative or non-zero	2(E2, E7)	X1	Amount entered by the user	E2: Amount is forwarded for further checking
					E7: Invalid amount
E2	Check withdrawal limit	2(E3, E7)	E1	Amount entered by the user	E3: Amount is forwarded for further checking
					E7: Invalid amount
E3	Check if amount is a multiple of 100 or not	2(E4 E7)	E2	Amount entered by the user	E4: Amount is forwarded for further checking
					E7: Invalid amount
E4	Check today‘s withdrawal limit	2(E5, E7)	E3	Amount entered by the user	E5: Amount is forwarded for Further checking
					E7: Invalid amount
E5	Check bank balance availability	2(E6, E7)	E4	Amount entered by the user	E6: Amount is checked
					E7: Invalid amount
E6	Granting withdrawal request	1(Z1)	E5	Remaining balance after withdrawal operation	Z: Remaining Balance Message print
E7	Not Granting withdrawal request	1(Y1)	E1, E2, E3, E4, E5	Invalid amount entered by the user	Y1: Error message displayed

3.2 Conversion of sequence diagram to sequence diagram graph

Sequence diagram explains how the objects interact with each other for a particular test scenario. Fig. 2 represents the SCSEDG for the withdrawal task of an ATM. Table 2 represents the dependency table for the withdrawal operation of ATM which is shown in the sequence diagram graph.

3.3 Generation and optimization of test cases

After generating SCSEDG graph, next stage generates and improve the TCs. The existing method uses several meta-heuristic methods for optimization, but in proposed method, for optimizing the TCs, the GWFF algorithm is applied. The test coverage criteria are calculated through TCs which covered a number of elements and the generation of TCs are reduced. The case generation using proposed method architecture is presented in Fig. 3.

At first, the population size and TC generations or number of iterations are fixed by the user. Then, a preliminary population is arbitrarily generated and their consistence fitness values are estimated and stored. The best initial optimal values are estimated. After that, the candidate solutions are ordered based on their fitness values. The maximum fitness values represent the solutions nearer to the optimality. After arranging the operation, bottom half of the poor solutions are rejected and uses the first half of the best solutions. These solutions undergo two various phases of optimization methods like GW and FF algorithms.

Phase 1: Grey-Wolf-Optimization

In GWO, the optimization is done by meta-heuristic techniques, which is bio-inspired from nature of grey wolves. In a grey wolf community there are four categories of grey wolves namely alpha, beta, delta, and omega [20]. Among that alpha is considered to be the leader of the group. Beta wolves assist alpha in decision making and hunting which are considered to be the next eligible candidate for an alpha, if alpha attains the stage of retirement or death while hunting. Delta wolves are elder wolves or former alpha wolves or sentinels or scout that protects the boundaries of their group. Omega wolves are the least prioritized wolves which need to submit their prey to all other wolves and follow all other category wolves.

Assume that every wolf is searching solution in the search space. The $w_i = \langle w_{i1}, w_{i2}, \dots \dots w_{in} \rangle$ represents position vectors in the search space, whereas the dimension of the problem is shown as n. The fitness function (based on problem definition) is employed to estimate the position of the wolves. Based on the fitness value the best wolves are classified as first solution that is represented as α, second is β, and third is δrespectively. In the best solution searching process, the wolves update their position according to the position of $\alpha, \beta \; and \; \delta$. In the starting stage, the wolf population is generated and the position of every wolf is initialized. After the initialization of the coefficients, every wolf (search agent) fitness value is estimated. After that, best fitness solutions are selected as first, second and third such as α, β, and δ, respectively.

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\left(\overrightarrow{D_\alpha}\right), \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\left(\overrightarrow{D_\beta}\right), \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\left(\overrightarrow{D_\delta}\right) \quad (1)$$

$$\overrightarrow{x(t+1)} = \frac{\overrightarrow{X_1}+\overrightarrow{X_2}+\overrightarrow{X_3}}{3} \quad (2)$$

In each iteration of the algorithm, the wolf's position update based on the position of wolves α, β according to the Eq. (1), (2). On the basis of new positions, the value of the fitness function of wolves is calculated and $\alpha, \beta \; and \; \delta$ will be selected. The GWO algorithm is used for ranking up every host resource based on their efficiency and utilization. As a result, the makespan decrease gradually.

Furthermore, the swarm intelligent methods are usually employed to solve the optimization problems that don't have a leader to monitor the entire proceeding period. This limitation is resolved in GWO method; the grey wolves have individual leadership capacity. Moreover, this algorithm uses few parameters and implementation is simple.

Phase 2: Fire-fly algorithm

The behaviour of fireflies flying in the tropical summer sky is used to propose a new FireFly algorithm (FF). The basic characters of FF are searching a prey, communicating and finding a mate by using bioluminescence with flashing patterns. These natural properties of FF are used to implement various metaheuristic algorithms. In this paper, some characteristics of FF are idealized to develop a FF-inspired algorithm. For simplicity, only three rules were followed:

a) There is no problem arises in their sex, because the FF will be attracted to other FF (i.e., all the FF are unisex).
b) The brightness is directly proportional to the attractiveness of FF. For instance, the less flashing FF will move towards the brighter flashing FF. The brightness may be decreased because of increasing distance between the FF. When comparing the particular FF with less bright FF, this individual will move randomly in the space.

The analytical form of the cost function is related to the brightness of FF. In particular, the value of the cost function is directly proportional to the brightness for maximization problem. The other forms of fitness function related to the brightness of FF are defined in genetic algorithm.

The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined in Eq. (3).

$$x^{t+1}{}_i = x_i{}^t + \beta_0 e^{-\gamma r^2{}_{ij}}\left(x^t{}_i - x^t{}_j\right) + \alpha\varepsilon^t{}_i \quad (3)$$

Where β_0 is the attractiveness at $r = 0$, the second term is due to attraction, while the third term is randomization with vector of random variables ε_i being drawn from a Gaussian distribution. The distance between any two fireflies i and j at x_i and x_j can be the Cartesian distance $r_{ij} = \left\|x_i - x_j\right\|_2$ or the $I_2 - norm$.

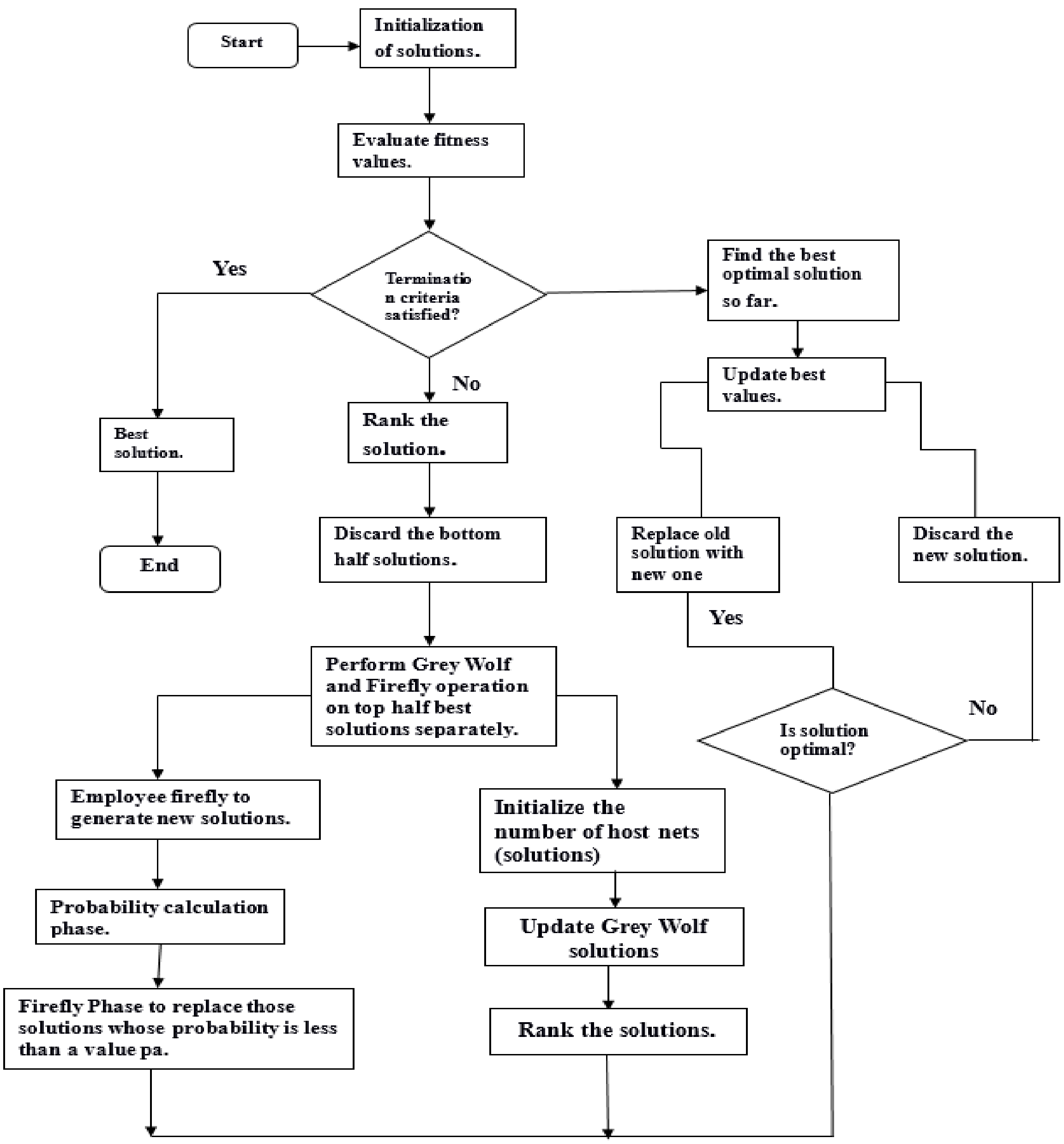
Start
Initialization of solutions.
Evaluate fitness values.
Termination criteria satisfied?
Yes
No
Find the best optimal solution so far.
Update best values.
Best solution.
End
Rank the solution.
Discard the bottom half solutions.
Replace old solution with new one
Discard the new solution.
Yes
Perform Grey Wolf and Firefly operation on top half best solutions separately.
Is solution optimal?
No
Employee firefly to generate new solutions.
Initialize the number of host nets (solutions)
Probability calculation phase.
Update Grey Wolf solutions
Firefly Phase to replace those solutions whose probability is less than a value pa.
Rank the solutions.

Figure.3 Basic structure of proposed methodology

4. Experimental result and discussion

The proposed GWFF technique was implemented in Net-Beans (version 8.2) on PC with 3.2 GHz with i5 processor. The proposed GWFF methodology is used for the generation of TCs with possible TCs from the combinational system diagram graph which combines the SCSEDG. The state chart diagram is under UML describe the time taken by a system software and this diagram consists of transitions of states. In our research, the withdrawal of ATM task is an example for chart diagram. The proposed method performance is measured using Mean Time Between Failures (MTBF).

MTBF: The MTBF includes the overall time period of TCs failure and overall time period to repair the TCs and its measure the software reliability. The MTBF mathematical description is presented in Eq. (4).

$$MTBF = \frac{1}{Probabilityoffailuresobtained} testcases \tag{4}$$

The Table 3 represents the fitness values and TCs / test data with various iterations. In this case, 200 iterations are considered. The function value depends upon the parametric values of the input variables. Fitness values of the proposed method GWFF are compared with the existing methods like Hybrid method i.e. Particle Swarm Optimization + Bee Colony Algorithm (PSBCA) [16] and BCA [17]. The existing method PSBCA is unable to handle the large programs for test case generations. In addition to this, the method BCA obtained the optimal value at 160 iterations, whereas the PSBCA reached 120 iterations for optimal solutions. The proposed GWFF method found that the solution reached its optimum value after 90 iterations.

The TC/test data of the proposed GWFF are tabulated in Table 4 in terms of maximum fitness values and the results are shown in Fig. 4.

The Table 4 shows that around 45% of TCs or test data have the higher the fitness function f(x) value and lies in between 0.7 and 1.0 fitness range using PSBCA but in case of BCA, only 25% of TCs or test data are available within the fitness value between 0.7 and 1.0. Finally, the proposed GWFF achieved 70% of TCs or test data having higher function f(x) value and lies between 0.7 and 1.0. Table 4 describes the fitness values of both existing method and GWFF method. From the table, the results proved that the GWFF method achieved better fitness values than the PSBCA and BCA.

Table 3. Fitness functional values of each test cases or test data with iteration numbers

Iteration Number	BCA [17]	PSBCA [16]	GWFF
	TCs/test data	TCs/test data	TCs/test data
1	4000	4500	4800
20	7600	10100	13100
40	14600	19800	20000
60	20900	24400	26500
80	26600	31400	36600
90	29400	35300	44000
100	32300	38600	44000
120	39000	42800	44000
140	43000	43800	44000
160	44000	43900	44000
180	44000	44000	44000
200	44000	44000	44000

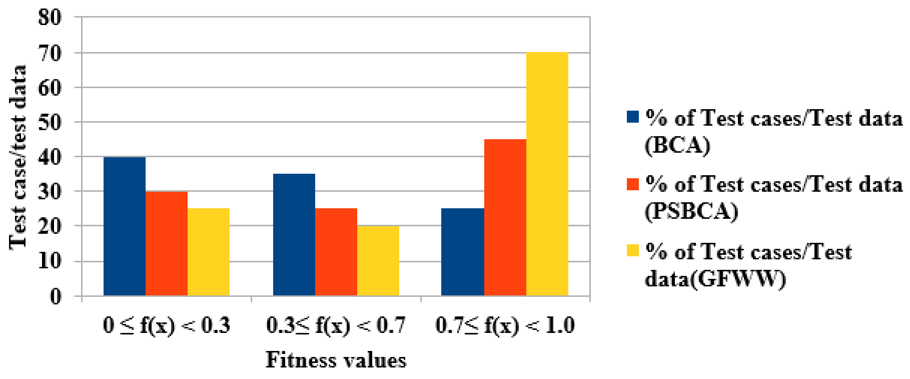

Figure.4 Percentage of TCs or test data vs fitness value ranges

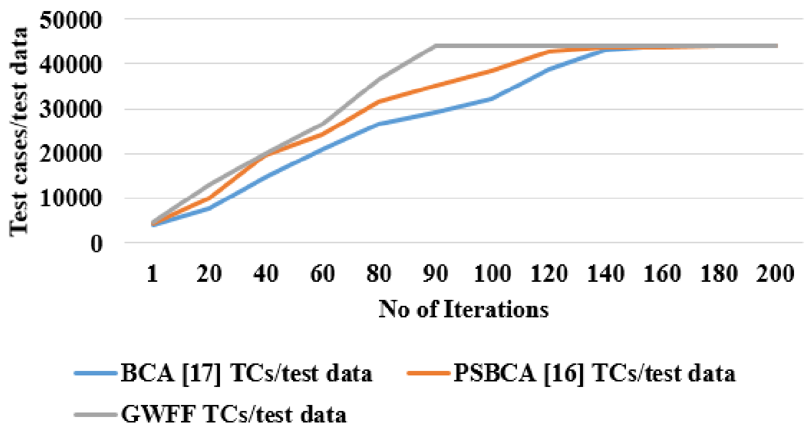

Figure.5 Different iteration vs TCs/Test data

Table 4. % of TCs or test data in terms of maximum fitness value

Fitness Value Range	% of TCs/Test data (BCA)	% of TCs/Test data (PSBCA)	% of TCs/Test data(GFWW)
0 ≤ f(x) < 0.3	40	30	25
0.3≤ f(x) < 0.7	35	25	20
0.7≤ f(x) < 1.0	25	45	70

The Fig. 4 represents the fitness value range with respect to the different number of TCs/test data. The proposed scheme obtains an automated TCs or test data belongs to the ATM withdrawal operation employing GWFF method. The Fig. 5 indicates the relation between two different variables such as different TCs and various iterations are shown in Table 3. Here, at 90th iterations the GWFF method attained an optimal solution approximately. The proposed scheme generated the TC or test data for Bank ATMs withdrawal operation using GWFF. The proposed method performance is presented in Table 5.

An experimental analysis of the proposed research work performance is measured through evaluation metrics such as MTBF and execution time. The proposed GWFF takes 18.2 Sec of execution time for selecting the best test path and its more capable, reliable for developing the software.

The Table 6 represents the comparative study of the ATM withdrawal operation based on different optimization techniques and proposed method's performance. All optimization methods are used in a similar fitness value range but the percentage of test data are different. Compare to all existing methods, the proposed GWFF algorithm achieved 70% of TCs/ test data having a high fitness function value and lies between 0.7 and 1.0.

5. Case study of withdrawal task of an ATM machine

The generated TC paths include a set of nodes which is the subset of the original set of nodes. By using SCSEDG system graph, the GWFF algorithm finds out the seven major possible traversing path. In all seven paths, only one path can generate an optimal result and remaining six paths doesn't obtain the optimal result.

Path 1: X -> A -> B -> D -> F -> Y
Path 2: X -> A -> B -> C -> E -> E1 -> E7 -> F -> Y
Path 3: X -> A -> B -> C -> E -> E1 -> E2 -> E7 -> F ->Y
Path 4: X -> A -> B -> C -> E -> E1 -> E2 -> E3 -> E7 -> F -> Y
Path 5: X -> A -> B -> C -> E -> E1 -> E2 -> E3 -> E4 ->E7 -> F -> Y
Path 6: X -> A -> B -> C -> E -> E1 -> E2 -> E3 -> E4 ->E5 -> E7 -> F -> Y
Path 7: X -> A -> B -> C -> E -> E1 -> E2 -> E3 -> E4 -> E5 -> E6 -> G -> Z

The path number represents as Path 1, Path 2, Path 3, Path 4, Path 5, and Path 6. All six paths give the improper optimal result and these paths are unsuccessful to achieve the ATM withdrawal operation. Only path 7 gives the proper optimized solution and it showed successful withdrawal operation.

Table 5. Performance of proposed GWFF method

Methodologies	MTBF	Execution Time
BCA	-	-
PSBCA	-	-
GWFF	48msec	18.2

Table 6. Bank ATM withdrawal operation based optimization techniques

Methodology Employed	Fitness Value Range	% of Test Cases/Test Data
PSO and BCA [16]	0 ≤ f(x) < 0.3	30
	0.3≤ f(x) < 0.7	25
	0.7≤ f(x) < 1.0	45
CS Algorithm [17]	0 ≤ f(x) < 0.3	30
	0.3≤ f(x) < 0.7	15
	0.7≤ f(x) < 1.0	55
Bee Colony Bat Algorithm [18]	0 ≤ f(x) < 0.3	35
	0.3≤ f(x) < 0.7	25
	0.7≤ f(x) < 1.0	45
Harmony search meta-heuristic search Technique [19]	0 ≤ f(x) < 0.3	25
	0.3≤ f(x) < 0.7	55
	0.7≤ f(x) < 1.0	20
Proposed GWFF	0 ≤ f(x) < 0.3	25
	0.3≤ f(x) < 0.7	54
	0.7≤ f(x) < 1.0	70

Table 7. Path movement through different nodes in ATM withdrawal operation employing SCSEDG

<Path 1	<Path 2	<Path 3	<Path 4	<Path 5	<Path 6	<Path 7
State X	State X	State X	State X	State X	State X	State X
A(m1,a,b)	A(m1,a,b)	A(m1,a,b)	A(m1,a,b)	A(m1,a,b)	A(m1,a,b)	A(m1,a,b)
B(m2,b,c)	B(m2,b,c)	B(m2,b,c)	B(m2,b,c)	B(m2,b,c)	B(m2,b,c)	B(m2,b,c)

D(m4,b,c)	C(m3,c,b)	C(m3,c,b)	C(m3,c,b)	C(m3,c,b)	C(m3,c,b)	C(m3,c,b)
F(m13,c,d)	E(m5,b,c)	E(m5,b,c)	E(m5,b,c)	E(m5,b,c)	E(m5,b,c)	E(m5,b,c)
State Y >	E1(m5,b,c)	E1(m5,b,c)	E1(m5,b,c)	E1(m5,b,c)	E1(m6,b,c)	E1(m6,b,c)
	E7(m12,b,c)	E2(m6,b,c)	E2(m6,b,c)	E2(m6,b,c)	E2(m7,b,c)	E2(m7,b,c)
	F(m13,c,d)	E7(m12,b,c)	E3(m7,b,c)	E3(m7,b,c)	E3(m8,b,c)	E3(m8,b,c)
	State Y>	F(m13,c,d)	E7(m12,b,c)	E4(m8,b,c)	E4(m9,b,c)	E4(m9,b,c)
		State Y>	F(m13,c,d)	E7(m12,b,c)	E5(m10,b,c)	E5(m10,b,c)
			State Y>	F(m13,c,d)	E7(m12,b,c)	E6(m11,c,b)
				State Y>	F(m13,c,d)	G(m14,b,d)
					State Y>	State Y>

This case study represents the model-driven generation of TCs or test data for large programs. The GW optimization is used to find the path very efficiently and quickly. This proposed method enhanced the efficiency and produced the optimized TCs.

6. Conclusion

In a model-driven approach based automated software testing, TCs are very useful. In this work, an evolutionary meta-heuristic algorithm called GWFF was proposed. The automated TCs was optimized with the test data by using this GWFF algorithm. This algorithm was used to generate the TCs which were optimized by taking an example of withdrawal operation through an ATM machine automatically. Test data values were selected based on the fitness function. This proposed approach optimized the TCs that were maximized with minimum iterations and time. An experimental analysis demonstrated that the GWFF method takes 16.4 Sec for the generation of path coverage. Compare to all existing methods, the proposed GWFF algorithm achieved 65% of TCs/ test data having higher fitness function value and lies between 0.7 and 1.0. The proposed GWFF method gave better results compared to PSO, CS, and BCA. In future, generate the path coverage by using different optimization technique to minimize the execution time better than the present work with the help of different case study.

References

[1] C. Jin, and S.W. Jin, "Prediction approach of software fault-proneness based on hybrid artificial neural network and quantum particle swarm optimization", *Applied Soft Computing*, Vol.35, pp.717-725, 2015.

[2] Y. Shi, M. Li, S. Arndt, and C. Smidts, "Metric-based software reliability prediction approach and its application", *Empirical Software Engineering*, Vol.22, No.4, pp.1579-1633, 2017.

[3] S.K. Dubey, and B. Jasra, "Reliability assessment of component based software systems using fuzzy and ANFIS techniques", *International Journal of System Assurance Engineering and Management*, Vol.8, No.2, pp.1319-1326, 2017.

[4] Y. Abdi, S. Parsa, and Y. Seyfari, "A hybrid one-class rule learning approach based on swarm intelligence for software fault prediction", *Innovations in Systems and Software Engineering*, Vol.11, No.4, pp.289-301, 2015.

[5] C. Diwaker, P. Tomar, R.C. Poonia, and V. Singh, "Prediction of Software Reliability using Bio Inspired Soft Computing Techniques", *Journal of Medical Systems*, Vol.42, No.5, p.93, 2018.

[6] N. Khurana, R.S. Chhillar, and U. Chhillar, "A Novel Technique for Generation and Optimization of Test Cases Using Use Case, Sequence, Activity Diagram and Genetic Algorithm", *JSW*, Vol.11, No.3, pp.242-250, 2016.

[7] B.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*, Vol.19, No.2, pp.737-753, 2016.

[8] C. Mao, "Generating test data for software structural testing based on particle swarm optimization", *Arabian Journal for Science and Engineering*, Vol.39, No.6, pp.4593-4607, 2014.

[9] G. Kumar, and P.K. Bhatia, "Software testing optimization through test suite reduction using fuzzy clustering", *CSI transactions on ICT*, Vol.1, No.3, pp.253-260, 2013.

[10] A. Kaushik, S. Verma, H.J. Singh, and G. Chhabra, "Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm", *International Journal of System Assurance Engineering and Management*, Vol.8, No.2, pp.1461-1471, 2017.

[11] A. Zaryabi, and A.B. Hamza, "A neural network approach for optimal software testing and maintenance", *Neural Computing and Applications*, Vol.24, No.2, pp.453-461, 2014.

[12] R.K. Sahoo, D.P. Mohapatra, and M.R. Patra, "A Firefly Algorithm Based Approach for

Automated Generation and Optimization of Test Cases", *International Journal of Computer Sciences and Engineering*, Vol.4, No.8, pp.54-58, 2016.

[13] M. Khari, P. Kumar, D. Burgos, and R.G. Crespo, "Optimized test suites for automated testing using different optimization techniques", *Soft Computing*, pp.1-12, 2017.

[14] M. Boopathi, R. Sujatha, C.S. Kumar, and S. Narasimman, "Quantification of Software Code Coverage Using Artificial Bee Colony Optimization Based on Markov Approach", *Arabian Journal for Science and Engineering*, Vol.42, No.8, pp.3503-3519, 2017.

[15] B.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,* Vol.19, No.2, pp.737-753, 2016.

[16] R.K. Sahoo, S.K., Nanda, D.P. Mohapatra, and M.R. Patra, "Model Driven Test Case Optimization of UML Combinational Diagrams Using Hybrid Bee Colony Algorithm", *International Journal of Intelligent Systems and Applications,* Vol.9, No.6, pp.43, 2017.

[17] R.K. Sahoo, D.P. Mohapatra, and M.R. Patra, "Model Driven Approach for Test Data Optimization Using Activity Diagram Based on Cuckoo Search Algorithm", *I.J. Information Technology and Computer Science*, Vol.10, pp.77-84, 2017.

[18] R.K. Sahoo, D.P. Mohapatra, and M.R. Patra, "Automated Testing Approach for Generation and Optimization of Test Cases using Hybrid Bat Algorithm. *International Journal of Computer Applications*, Vol.161, No.7, 2017.

[19] R.K. Sahoo, D. Ojha, D.P. Mohapatra, and M.R. Patra, "Automatic generation and optimization of test data using harmony search algorithm", *Computer Science & Information Technology*, p.23, 2016.

[20] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer", *Advances in engineering software,* Vol.69, pp.46-61, 2014.