A STRATEGY FOR MUTATION TESTING USING GRAVITATIONAL SEARCH ALGORITHM

[1]Priyanka Dabur, [2]Rajvir Singh
[1]M.Tech. Scholar, Department of Computer Science Engineering, DCRUST, Murthal
[2]Assistant Professor, Department of Computer Science Engineering, DCRUST, Murthal

Abstract: ***Software testing is a process of execution of the program to detect the faults in software system. It is very time consuming process. There are number of techniques available to perform software testing, mutation testing is one of such testing techniques. In mutation testing, a minor semantic change is introduced to the original code. The effectiveness of different test cases is then compared using mutation score. In the proposed work, we have generated the mutants using pymute tool available in MATLAB. It creates the mutants by replacing the algebraic sign, changing the variables and conditions. These generated mutants are tested for deviation of result from the original system under test. The test is done for each mutants for optimization using gravitational search algorithm (GSA) and simple genetic algorithm (GA).We considered this problem for the nonlinear problem of optimization. Previously genetic algorithm was used for this purpose, but it stuck into global maxima for multi objective functions. So, GSA is used for global optimization.***

Keywords: Test case, Gravitational search algorithm, Mutation testing, optimization.

1. INTRODUCTION

Mutation testing, a method of software testing is used to test manipulated program and the original program. The source code is introduced with mutations to imitate common programming errors.In literature, various evolutionary algorithms like GAare used for mutation based test case generation and reduction. The Gravitational Search Algorithm (GSA) is one of the evolutionary search algorithms. GSA is heuristic optimization algorithm that attracts the attention of the researchers and is based on Newton's law of gravity and laws of motion. In this paper, a technique for test suite reduction is presented which uses gravitational search algorithm (GSA). The proposed approach allows to add the test cases which evaluate mutants with the maximum relative error first in the set of selected test cases i.e. reduced test suite. Test cases are evaluated using the ackley, bird, beale, cube functions as the fitness functions as also used in earlier works. As a final step, the ackley, bird, beale, cube functions are unit tested by using independent technologies to calculate expected outputs from the generated test inputs.Comparisons are drawn with the results from existing manual test cases generated using GA in literature. The technique improves the exploration of the global search space using the algorithm based on the gravity law.

2. LITERATURE SURVEY

R. H. Rosero. et.al [1] presented a technique based on selective regression testing which helps in minimizing the test suite.

M. B. Basher et.al [2] presented a technique based on genetic algorithm and considered a novel state- based, control-oriented fitness function for the research work which uses oops features to evaluate the test case efficiently.

Khan, R., &Amjad, M et.al [3] proposed a technique based on genetic algorithm with mutation analysis to increase the efficiency of software testing. They introduced mutants in original program and found the optimized test cases using genetic algorithm.

Ahlam Ansari, AnamKhanetet.al [4] adopted a different approach than optimization algorithm. They minimized the number of test cases by using finite automata, and Extended DFA. The method improved the accuracy in test cases generation by validation.And increased the efficiency by model combination.

To follow the legacy of test case prioritization, Muhammad Khatibsyarbini, MohdAdham [5] used ant colony optimization (ACO) to reduce the test cases by considering the issue as NP hard problem.

Masud, M., Nayak, A., Zaman, M., & Bansal, N.et.al [6] proposed a techniques based on genetic algorithm to evaluate faults and kill the mutant in test program.

Chu-Ti Lin et al. [7] presented the test case reduction method based on genetic algorithm and control flow and data flow-based test coverage criteria. The GA is used to automatically generate the test data for these spanning sets which minimized the test cases and automated the test path generation.

Jiang, W.K. Chan,et.al [8] proposed the test case reduction by using Analytical Hierarchy Process (AHP) and the Weighted Sum Model (WSM). The major focus of research in this study was the cost reduction along with the number of test case reduction. The WSM model is particularly used for low cost reduction in test case minimization. The results also showed that when using the method for reducing the cost then it not gave the same results.When WSM was evaluated, it proved to be the least cost-effective of the strategies.

Erik Rogstad, Lionel Briand, Richard Torkaretet. al [9] presented a cost effective algorithm for test case reductions. Their paper focused on the reduction of test cases targeted on cost reduction which was not discussed in many earlier researches. They used a cost-aware metric called

Irreplaceability, which checked the probability that each test case can be replaced by others when reducing test suites. Authors also tested the Irreplaceability and other test case metric 'ratio' with three well-known test suite reduction algorithms (i.e., Greedy, GRE, and HGS).

Ma, Yu-Seung, Yong-Rae Kwon, and Jeff Offuttet [10] advocateda method which decreased the search space for test cases and also reduced the cost. He used LBP method to reduce the test cases and uniformly spreader them into regression test cases. He controlled the space for local search to make the method more cost effective. Results proved that their proposed method was as effective as the best search based algorithms like greedy, hill climbing. The best technique is LBS110.Which achieves highest mean AFPD values than all the other techniques studied in the controlled experiment.

Black, Paul E., VadimOkun, and YaacovYeshaet [11] proposed the application of database reduction using black box technique and tested it under real industrial environment in Norway's tax department. They adopted the classification tree model in black box testing and suitable for those cases where source code analysis is not possible. They compared the similarity based method with random selection and random selection with partitions. It was proved that similarity based method as objective function for the heuristic optimization is more efficient with regards to fault detection rate.

Jia, Yue, and Mark Harmanet [12] proposed a method based on prioritization which considered several methods which affects the error propagation to successor. The work is done in two steps, in the first step classes are prioritized and in the second level the test cases of prioritized classes are ordered. The proposed algorithm is implemented in C++ and results are compared with previous work.

J. H. Andrews, L. C. Briand, and Y. Labicheet [13] presented the test case reduction and prioritization based on genetic algorithm. The main focus was to evaluate modification in a method's body because of data dependence, control dependence and dependenciesbecause of inheritance and polymorphism. To select the test cases based on affected statements, they ordered them based on their fitness by using GA.

Y. Jia and M. Harman [14] proposed the algorithm which is based on six factors: (1) customer priority, (2) changes in requirement, (3) implementation complexity, (4) requirement traceability, (5) execution time and (6) fault impact on requirements. Authors tested the algorithm on two datasets for early fault detection. Mean percentage of fault detection rate was considered as evaluation parameter.Results showed that proposed method is better than random prioritization method in generating the test case sequences for early fault detection.

A. Simao, J. C. Maldonado, and R. da Silva Bigonha[15] presented the mutation testing at code level using Simulink in MATLAB. Authors presented the search based method for generatingthe test data for mutants killing. It has been noticed that complexity introduced in the test case generation remain challenging, whatever be the abstraction level. The challenges are same whether the Simulink in MATLAB is used to generate the test cases or done at code level.

Mathuret.al [16] presented the graph based method foroptimized using GA similar to [2][4]. The method arranged the test cases based on their fitness value compared with the previous fault history. The results were evaluated on the basis of average percentage rate of fault detection (APFD). Their proposed approach provided the better results in terms of APFD.

King, Kim N., and A. Jefferson Offutt [17] used GA for optimizationneural network and fuzzy logic systemconsidering different types of test cases. They concluded that by using GA the results improved.Offutt, A. Jefferson [18]used the GA for test cases reduction.

Similar to the existing work, the proposed methodology in this paper presents the method for test case reduction using GSA and GA. And the results are compared and analyzed. The results showed that the GSA is better as compared to GA.

3. PROPOSED METHODOLOGY

The proposed methodology uses MAT mute software which requires the two tools Python 2.5 or 2.6 and MATLAB. To perform MAT mute function project directory is including in MATLAB path, and the Pymute directory should be added to the Python path. Pymute tool creates the mutants by replacing the algebraic sign, changing the variables and based on other conditions. These generated mutants are tested for relative error i.e. deviation of result from the actual test suit.

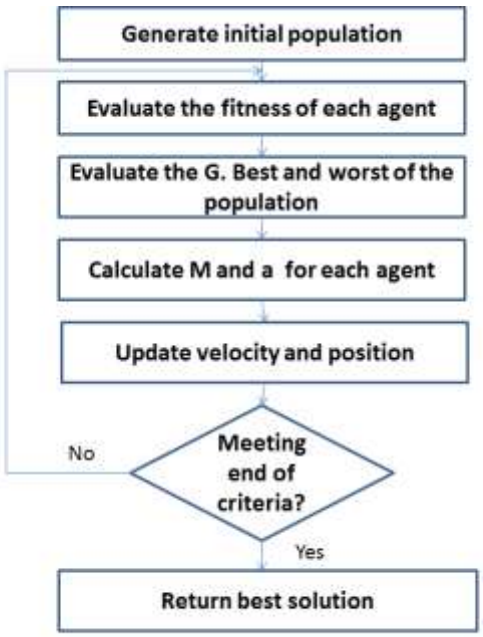

Fig 1: Flowchart of Gravitational search algorithm

The GSA follows the procedure mention below.

The GSA follows the procedure mentioned below:

1) Initialize domain.

2) Initialize the population randomly.

3) Calculate the fitness value for each agent.

4) Update G(t), best(t), worst(t) and Mi(t) for i = 1,2,.. .,N.

5) Evaluate total force in different directions.

6) Evaluate acceleration and velocity.

7) Update agent's position.

8) Repeat steps 3 to 7 until the stop criteria are reached.

9) End.

In previous works, genetic algorithm was used for the purpose of test suite reduction, but it stuck into local minima for multi objective functions.So,we used a better and global optimization method named gravitational search algorithm (GSA) in our proposed work.

During implementation, the programwas designed to generate mutants and optimize them to search for nearest mutant to actual codefor the MATLAB test functions. Benchmark functionsexisting in MATLAB library were used for optimization. Every mutant generated is tested with proposed methodology based on GSA.

4. DISCUSSION OF RESULTS AND ANALYSIS

Results are compared in terms of detection score vs. relative error for the benchmark functions as discussed below.

4.1 Case Study-I: Test Suite 1. Ackley Function

Ackley function has two input variables and one output. The test suite defined has lower and upper boundaries of the function. A total of 170 mutants are generated by pymute tool for test function. Each mutant is tested with GSA and GA separately. Figure 2 shows the detection score comparison for both methodsi.e using GA and using GSA. The y-axis in the graph shows the percentage of detected mutantsas rest mutants are having infinite error or zero error. We dropped those mutants and considered rest only.

On the x- axis the relative error is shown. The graph shows that with more number of mutants the proposed method performs better. In our proposed method using GSA more are the mutants detectedas compared with the existing methods using GA as shown in the graph in figure 2. The minimum detection scale is 30% for GSA and for GA it is less than 20%.

As a result, out of 170 total mutants, GSA detected 108 and GA detected 99 mutants.

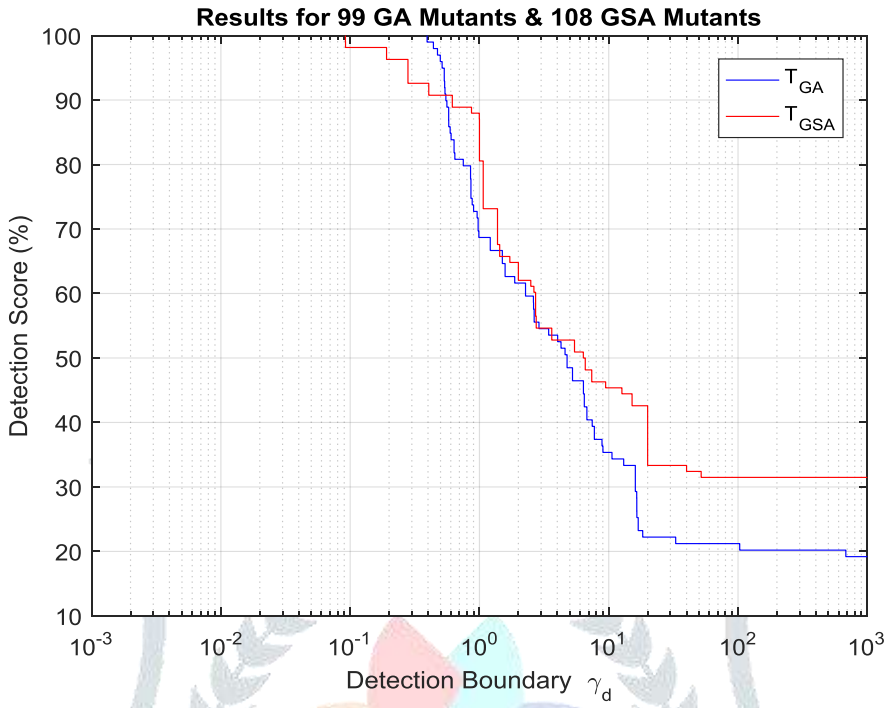

Figure 2: Detection score for GSA and GA for ackley test suite function

The benchmark optimization test suit functions are looking for a global minima point and maxima. We plotted a 3D surface view for all 170 mutants.Finally, tuned set of values for GSA tunedackley function in figure 3(a) and GA tuned ackley function in figure 3(b) are shown. From figure 3(a) and 3(b), it can be seen that the GSA tuned set of input variables to mutants generated the global minima point nearer to minima point generated by original ackley test suite function as compared to GA tuned ackley test suite function.

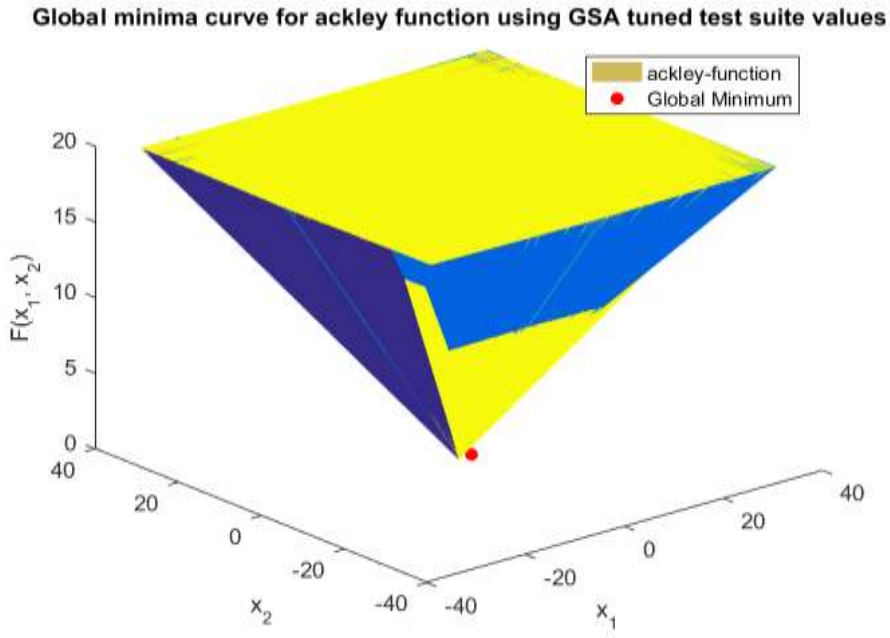

Figure 3(a): 3D global minima plot for GSA tuned input pair fororiginal ackley function for 170 mutants

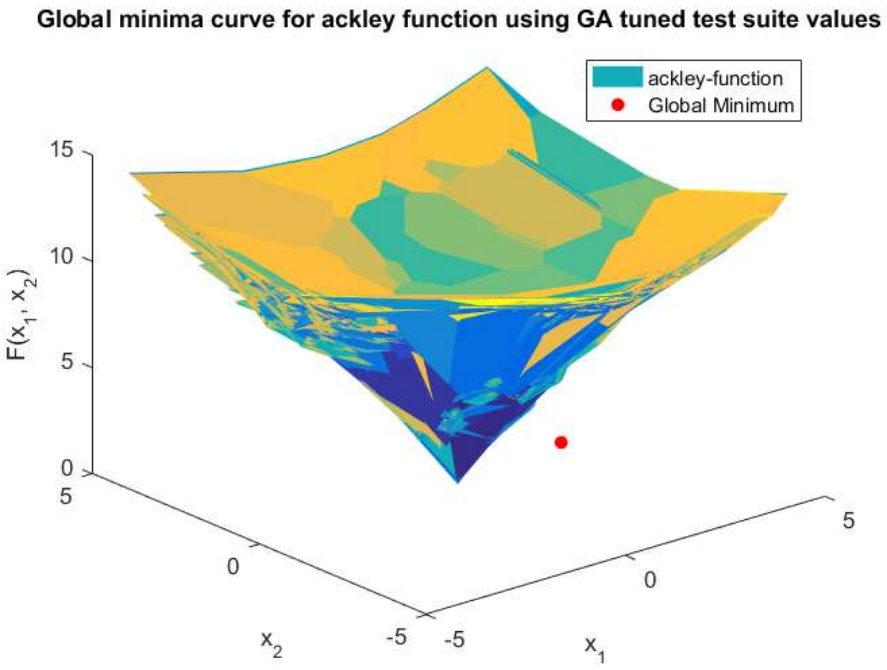

Figure 3(b): 3D global minima plot for GA tuned input pair for ackley function for 170 mutants

Table.1. The analysis table for GA and GSA tuned Ackley function for mutation testing (Total Mutants=170).

	Proposed GSA based method	Existing GA based method
Number of Detected Mutants	108	99
Error	0.0919	0.395

4.2 Case Study-II: Test Suit 2. Bird test suite

The detection score for the bird function is shown in figure 4. It has the 247 mutants generated by pymute with 2 input variables. These mutants are reduced to 88 and 90 by GA and GSA respectively.

The GSA detection score is high than GA and graph is moving towards right direction which is desirable. Though the error for 100% detection in GSA is more than GA but total detection score is better in GSA.

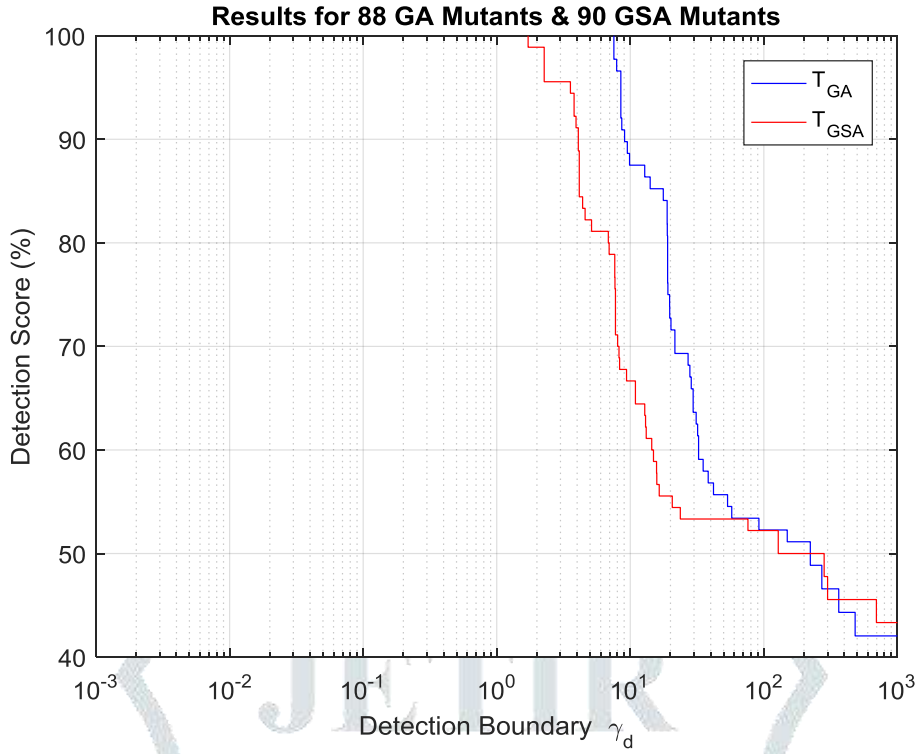

Figure 4: Detection score for GSA and GA for bird test suite function

The global minima 3D plots for bird test function is shown in figure 5(a) and 5 (b) like previous test suits.

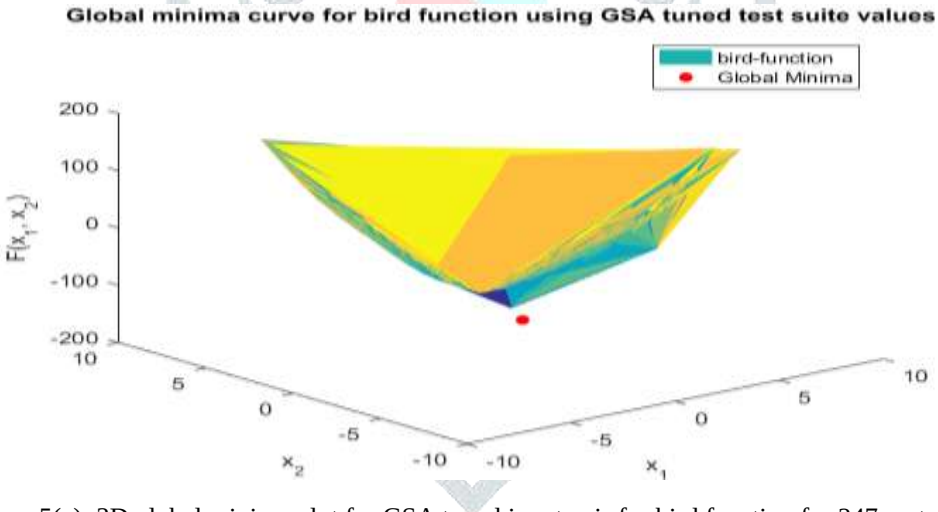

Figure 5(a): 3D global minima plot for GSA tuned input pair for bird function for 247 mutants

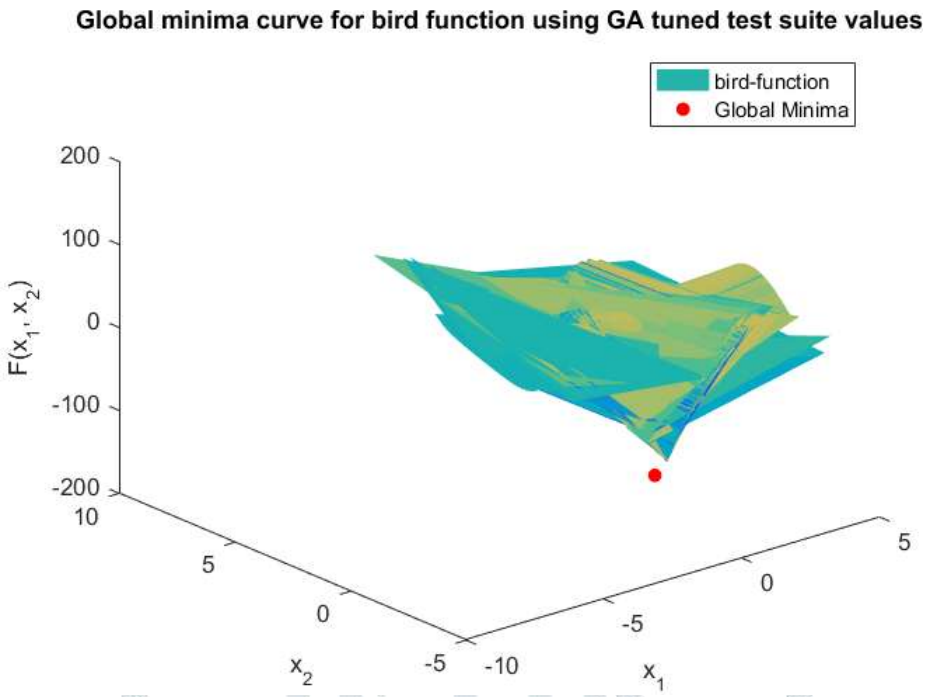

Figure 5(b): 3D global minima plot for GA tuned input pair for bird function for 247 mutants.

The Table 2 below shows the comparison of GSA over GA mutation testing score using bird function.

Table.2. The analysis table for GA and GSA tuned Ackley function for mutation testing (Total Mutants=247).

	Proposed GSA based method	Existing GA based method
Number of Detected Mutants	90	88
Minimum Error	1.71	7.53

4.3 Case Study-III: Test Suite III. Beale Function

Detection score for the beale function is shown in figure 6. It has the 177 mutants generated by pymute with 2input variables. The mutants are reduced to 124 by GA and GSA respectively.

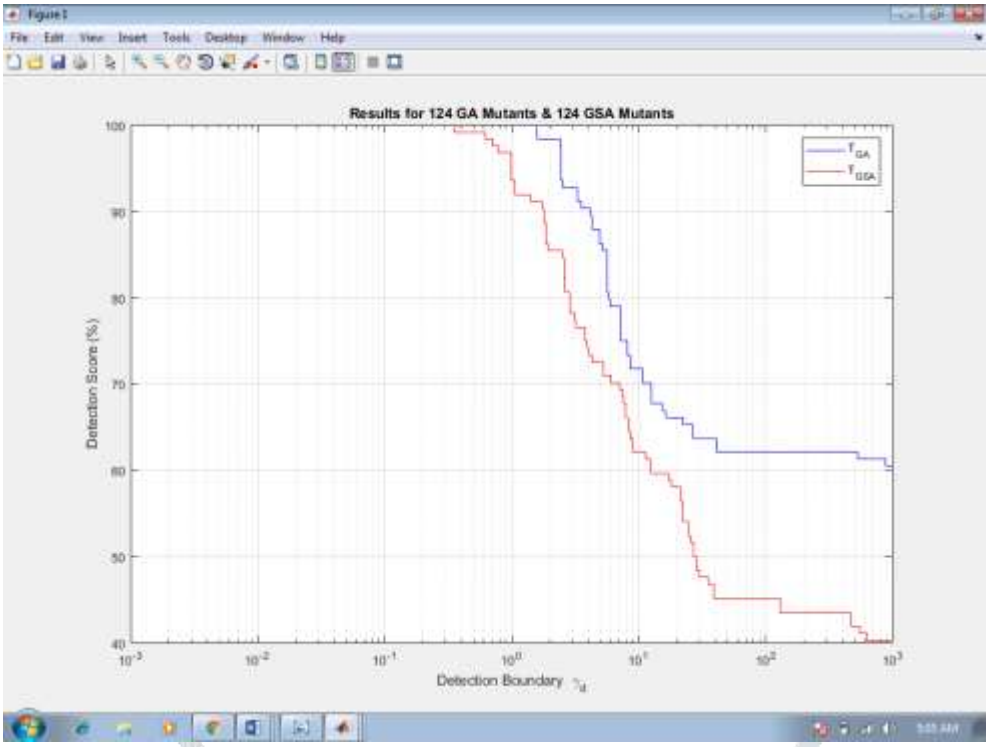

Figure 6: Detection score for GSA and GA for beale test suit function

The global minima 3D plots for beale test function are shown in figure 7(a) and 7(b) like previous test suits.

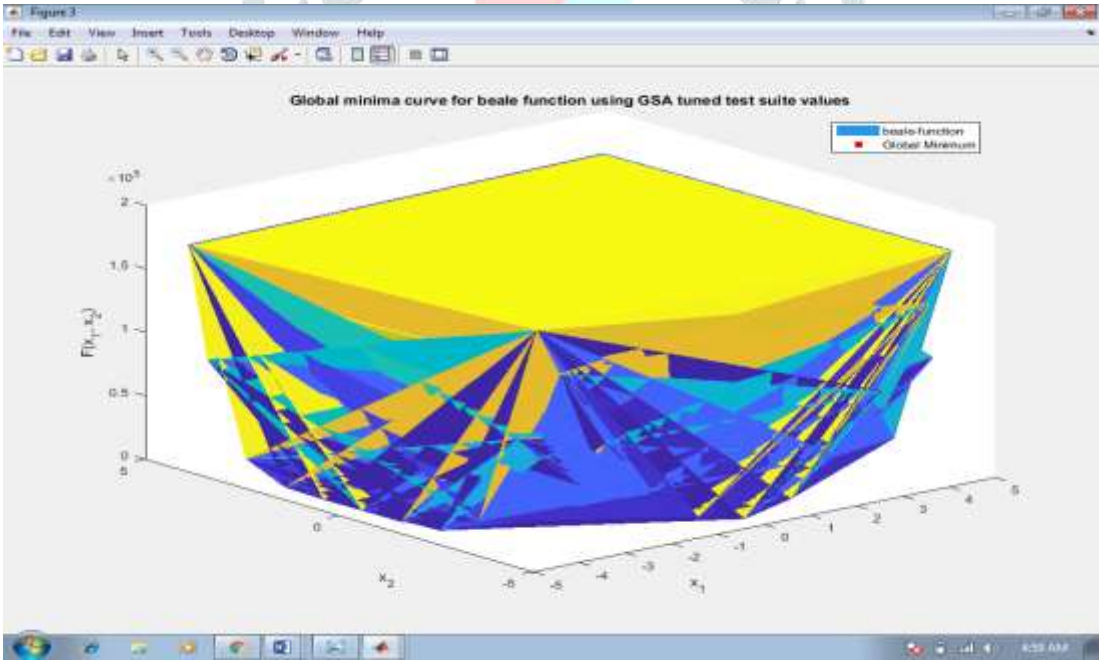

Figure 7(a): 3D global minima plot for GSA tuned input pair for beale function for 177mutants

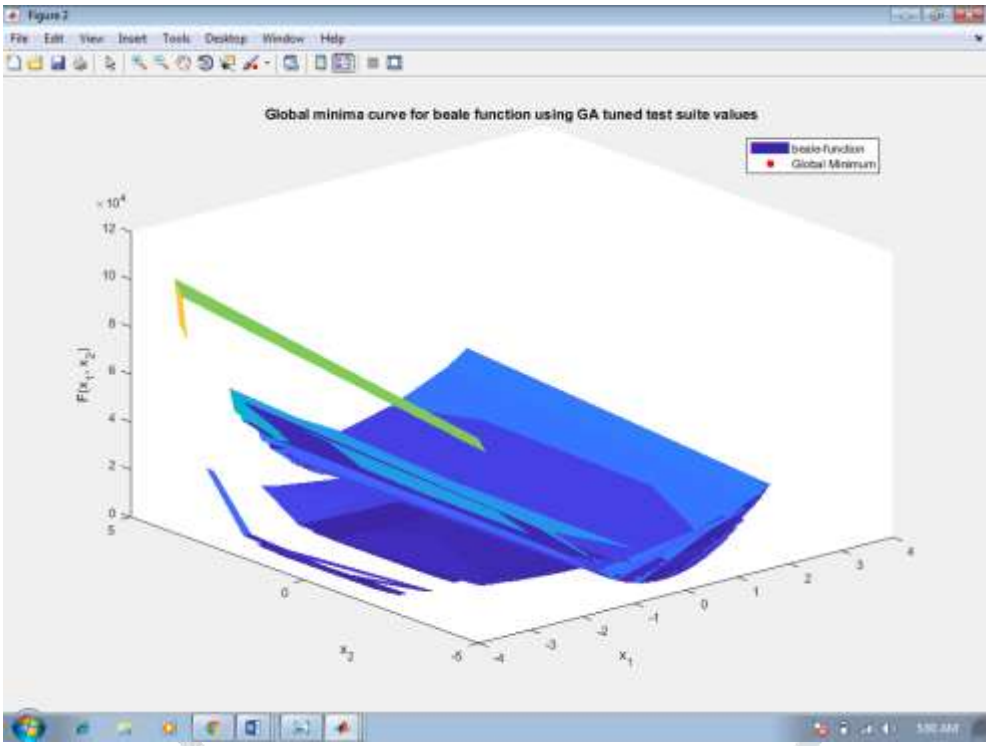

Figure 7(b): 3D global minima plot for GA tuned input pair for beale function for 177 mutants

The Table 3. Shows the comparison of GSA over GA mutation testing.

Table 3.The analysis table for GA and GSA tuned Ackley function for mutation testing (Total Mutants=124).

	Proposed GSA based method	Existing GA based method
Number of Detected Mutants	124	124
Minimum Error	1.71	7.53

5. CONCLUSION AND FUTURE SCOPE

GSA has been applied for reduction of test suite for mutation based testing. GSA had been implemented to find optimized test cases with the help of mutation analysis, where we applied three benchmark functions to calculate the mutation score. The mutants are injected in the program and random test cases are generated and mutation score is calculated accordingly the test cases are selected based on mutation detection score i.e. test suite is reduced. The results showed that the GSA based proposed methodology in this paper is better as compared with GA based methodology in literature.

In future, the cases of infinity mutants may be tried or analyzed which are ignored in this paper. Further, the hybrid techniques with GSA using fuzzy logic and neural network for the test case generationmay be tried.

REFERENCES

[1]. R. H. Rosero, O. S. Gómez and G. Rodríguez, "Regression Testing of Database Applications Under an Incremental Software Development Setting," in *IEEE Access*, vol. 5, pp. 18419-18428, 2017.
[2]. M. B. Bashir and A. Nadeem, "Improved Genetic Algorithm to Reduce Mutation Testing Cost," in *IEEE Access*, vol. 5, pp. 3657-3674, 2017.
[3]. Khan, R., &Amjad, M. (2016, March). Optimize the software testing efficiency using genetic algorithm and mutation analysis. In *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on* (pp. 1174-1176). IEEE.C. Zhang, Z. Duan, B. Yu, C. Tian and M. Ding, "A Test Case Generation Approach Based on Sequence Diagram and Automata Models," in *Chinese Journal of Electronics*, vol. 25, no. 2, pp. 234-240, 3 2016.
[4]. Ahlam Ansari, Anam Khan, Alisha Khan, KonainMukadam, Optimized Regression Test Using Test Case Prioritization, Procedia Computer Science, Volume 79, 2016
[5]. Muhammad Khatibsyarbini, MohdAdham Isa, Dayang N.A. Jawawi, Rooster Tumeng, Test case prioritization approaches in regression testing: A systematic literature review, Information and Software Technology, Volume 93, 2018

[6]. Masud, M., Nayak, A., Zaman, M., & Bansal, N. (2005, May). Strategy for mutation testing using genetic algorithms. In *Electrical and Computer Engineering, 2005. Canadian Conference on* (pp. 1049-1052). IEEE.Amanda Schwartz, Hyunsook Do, Cost-effective regression testing through Adaptive Test Prioritization strategies, Journal of Systems and Software, Volume 115, 2016

[7]. Chu-Ti Lin, Kai-Wei Tang, Gregory M. Kapfhammer, Test suite reduction methods that decrease regression testing costs by identifying irreplaceable tests, Information and Software Technology, Volume 56, Issue 10, 2014

[8]. Jiang, W.K. Chan, Input-based adaptive randomized test case prioritization: A local beam search approach, Journal of Systems and Software, Volume 105, 2015

[9]. Erik Rogstad, Lionel Briand, Richard Torkar, Test case selection for black-box regression testing of database applications, Information and Software Technology, Volume 55, Issue 10, 2013,

[10]. Ma, Yu-Seung, Yong-Rae Kwon, and Jeff Offutt. "Inter-class mutation operators for Java." Software Reliability Engineering, 2002.ISSRE 2003.Proceedings.13th International Symposium on.IEEE, 2002.

[11]. Black, Paul E., VadimOkun, and YaacovYesha. "Mutation operators for specifications."Automated Software Engineering, 2000.Proceedings ASE 2000.The Fifteenth IEEE International Conference on.IEEE, 2000.

[12]. Jia, Yue, and Mark Harman. "An analysis and survey of the development of mutation testing." Software Engineering, IEEE Transactions on 37.5 (2011): 649-678.

[13]. J. H. Andrews, L. C. Briand, and Y. Labiche. Is mutation an appropriate tool for testing experiments? In ICSE '05: Proceedings of the 27th international conference on Software engineering, pages 402–411, New York, NY, USA, 2005. ACM

[14]. Y. Jia and M. Harman, "MILU: A Customizable, Runtime-Optimized Higher Order Mutation Testing Tool for the Full C Language," in Proceedings of the 3rd Testing: Academic and Industrial Conference Practice and Research Techniques (TAIC PART'08). Windsor, UK: IEEE Computer Society, 29-31 August 2008, pp. 94–98.

[15]. A. Simao, J. C. Maldonado, and R. da Silva Bigonha, "A Trans-formational Language for Mutant escription,"ComputerLanguages,Systems& Structures, vol. 35, no. 3, pp. 322–339, October 2009.

[16]. Mathur, Aditya P. "Performance, effectiveness, and reliability issues in software testing." Computer Software and Applications Conference, 1991.COMPSAC'91.,Proceedings of the Fifteenth Annual International. IEEE,1991.

[17]. King, Kim N., and A. Jefferson Offutt. "A fortran language system for mutation-based software testing." Software: Practice and Experience 21.7 (1991): 685- 718.

[18]. Offutt, A. Jefferson. "Investigations of the software testing coupling effect." ACM Transactions on Software Engineering and Methodology (TOSEM) 1.1 (1992): 5-20.