Comparative Analysis of MOGA, NSGA-II and MOPSO for Regression Test Suite Optimization

Article · January 2014

CITATIONS
10

READS
1,031

2 authors:

Zeeshan Anwar
National University of Sciences and Technology
22 PUBLICATIONS **47** CITATIONS
SEE PROFILE

Ali Ahsan
Torrens University Australia
63 PUBLICATIONS **94** CITATIONS
SEE PROFILE

Comparative Analysis of MOGA, NSGA-II and MOPSO for Regression Test Suite Optimization

Zeeshan Anwar(1) and Ali Ahsan(2)

(1) Center for Advance Studies in Engineering (CASE), Islamabad (Pakistan)
E-mail: zeeshan0333@yahoo.com

(2) Center for Advance Studies in Engineering (CASE), Islamabad (Pakistan)
E-mail: al_ahsan1@yahoo.com

ABSTRACT

In Software Engineering Regression Testing is a mandatory activity. Whenever, a change in existing system occurs and new version appears, the unchanged portions need to be regression tested for any resulting undesirable effects. During process of Regression Testing, same test cases are executed repeatedly for un-modified portion of software. This activity is an overhead and consumes huge resources and budget. To save time and resources, researches have proposed various techniques for Regression Test Suite Optimization. In this research regression test suites are minimized using three Computational Intelligence multi-objective techniques for black box testing methods. These include; 1- Multi-Objective Genetic Algorithms (MOGA), 2- Non-Dominated Sorting Genetic Algorithm (NSGA-II) and 3- Multi-Objective Particle Swarm Optimization (MOPSO). Said techniques are applied on two published case studies and through experimentation, the quality of these techniques is analyzed. Four quality metrics are defined to perform this analysis. The results of research show that MOGA is better for reducing the size and thus execution time of the regression test suites as compared to MOPSO and NSGA-II. It was also found that use of MOGA, NSGA-II and MOPSO are not safe for regression test suite optimization. This is because fault detection rate and requirement coverage is reduced after optimization of Regression Test Suites.

Keywords—Regression Testing, Regression Test Suite Optimization, Computational Intelligence, Multi-Objective Genetic Algorithm, Non-Dominated Sorting Genetic Algorithm, Multi-Objective Particle Swarm Optimization.

1- INTRODUCTION

Whenever software is modified, new test cases are written to test this modification. These new cases are then added into the existing test suites the size of which thus increases. In this way, with time, some of the old test cases become obsolete, broken and redundant and it becomes too expensive to run the whole test suite for every change. This ever-increasing size of test suites requires it to be optimized in order to reduce its size and minimize the execution time [1]. Researchers have optimized regression test suites using minimization, ranking or prioritization, and selection techniques. This is done using various heuristic, search based, and computational intelligence based methods. Most of

the existing methods for regression test suites optimization are static and single-objective[1]. For this reason, optimization of the selected test suite is avoided, because during optimization, many important test cases may be skipped resulting in an unsafe optimization approach. [2] Testing is a multi-objective activity because one has to cater for many objectives like reducing time, maximizing fault detection rate coverage. Hence, multi-objective[2] optimization approach is required for making optimization safe [3].

Researchers proposed Multi-Objective Evolutionary Algorithms (MOEA) [4] for this purpose and most of the research on regression test suite optimization is done on code coverage based techniques. However, in many cases, code is not available and black box testing is performed. In the existing literature, the authors did not find any study that solved the regression test suite optimization problem for black box testing using MOGA, NSGA-II and MOPSO algorithms. Researchers have also limited their scope to three objectives. This paper, however, proposes four objectives for regression test suite optimization.

1-1 SCOPE OF WORK

In this research, the process of regression testing was studied the problem was formulated as multi-objective optimization problem for Black Box Testing. Regression test suite optimization problem is solved using the computational intelligence techniques (MOGA, NSGA-II and MOPSO).

1-2 SIGNIFICANCE OF THE STUDY

This study explores the use of those optimization techniques on black box testing which have previously only been used for white box based regression test suites optimization. This work thus opens a new area for researchers. MOGA, NSGA-II and MOPSO have never been used before for black box based regression test suites optimization. The objectives of optimization of regression test suites are identified and the optimization is then simulated using MOGA, NSGA-II and MOPSO. Finally, results of the simulated techniques are compared and analyzed. In the existing literature, the comparison of these techniques cannot be found.

2- LITERATURE REVIEW

2-1 REGRESSION TEST SUITE OPTIMIZATION

Whenever bugs are fixed, new features are added or old features are removed, one has to add new test cases into the existing test suite. However, old test cases may not be useful because one has to change the existing test cases when bugs are fixed. In case of addition of new features one has to add new test cases and in case of removing old features one has to remove test cases

[1] Single objective optimization considers only one objective at a time; it is not a safe technique for regression testing.
[2] Multi-objective optimization considers many objective of interest at a time to find best solution.

that are related to old features and add new test cases to check that removal of old features does not affect the unmodified features. Test cases can be divided into five classes. Reusable test cases are used for the unchanged parts of programs and these test cases are unnecessary but can be used for future versions of programs. Retestable test cases are for changed parts of programs they must be retested. Obsolete test cases do not contribute to the testing of modified program because input / output or structure of program is modified but still they are present in the test suite. To test the modified structure of program new structural test cases are written these test cases also test the changed specifications [5].

Five classes of test cases are present in the test suite and many of these test cases did not contribute in the testing of system. These test cases just increase the size of test suite and make it ineffective. To make the regression test suite efficient to save time, cost or resources is regression test suite optimization. Optimization is required because size of test suite is very large and we always have time, budget and resource constraints and testing the whole test suite is not possible even for small modifications therefore one has to optimize regression test suite [6]-[8].

2-2 TECHNIQUES OF REGRESSION TEST SUITE OPTIMIZATION

Different techniques for regression testing have been defined in the literature. Retest all, Regression Test Selection, Test Case Prioritization and Hybrid approach are defined by Gaurv Duggal and Bharti Suri [9]. Similarly Graves et al identified five techniques for regression testing namely Minimization Techniques, Data Flow Techniques, Safe Techniques, Adhoc / Random Techniques and Retest All Techniques [10]. Regression Test Suite Optimization can be applied on the three approaches of regression testing and is used to minimize, prioritize or select the test cases from the test suite based on the desired parameters or objective functions. These techniques are used in different ways and with combination of each other. Wong et al used minimization and prioritization to select regression test suites [6]. Malhotra et al used modification based test suite prioritization technique for test case selection [7].

2-3 COMPUTATIONAL INTELLIGENCE (CI)

Biologically inspired algorithms are used in Computational Intelligence to solve problems. Neural Networks, Connectionist Systems, Genetic Algorithms, Evolutionary Programming, Fuzzy System and Hybrid Intelligent Systems are techniques of Computational Intelligence.

S. Sumathi and Surekha P. defined computational intelligence as successor of artificial intelligence. CI gets biological inspirations to solve complex problems and those problems for which there is no computational solution. CI can be divided into primary branches, other approaches and Hybrid approaches. Neural Network, Fuzzy Systems, Swarm Intelligence and Evolutionary Computing are four primary branches of CI. Granular Computing, Chaos Theory and Artificial Immune Systems fall in category of other approaches of CI. Neural Networks controlled by Fuzzy Logic, Fuzzy Logic Controllers tuned by Neural Networks, Neuro Fuzzy Systems, Neural Networks generated by Genetic Algorithms, Genetic Algorithms controlled by Fuzzy Logic, Fuzzy Logic Controllers tuned by

Genetic Algorithms, Fuzzy Logic Controllers' learning optimization by Genetic Algorithms, Fuzzy Evolutionary Systems and Fuzzy Logic controllers generated by Genetic Algorithms are Hybrid Approaches of CI. CI approaches can be used to solve five classes of problems namely Optimization Problems, Classification Problems, Control Problems, Regression Problems and NP Complete Problems. Detailed description of these problems along-with approaches can be found at [11].

2-4 USE OF CI APPROACHES FOR REGRESSION TEST SUITE OPTIMIZATION

CI algorithms are being used in literature to solve the complex problems of software engineering, similarly for regression test suite optimization following approaches of CI are used frequently.

2-4-1 RTO AND GENETIC ALGORITHMS

Single objective and multi-objective genetic algorithms are used for the optimization of regression test suite. Notable work for said purpose is work by Yoo et al, who used Pareto efficient multi-objective test suite selection. This work can be divided into two versions. In first version coverage and cost whereas in second version coverage, cost & faults history are taken as objectives. Authors compared greedy, additional greedy, NSGA-II & vNSGA-II algorithms for both versions. Pareto front, Pareto efficient and t-test are used for analysis. It is proposed that Greedy algorithm performs well for single objective but not always efficient in multi-objective paradigm [12].

Li et al used five algorithms Greedy, Additional Greedy, 2-Optimal, Hill Climbing and Genetic Algorithms for regression test suites prioritization and applied these algorithms on six programs to optimize test suites. Coverage based objectives (block, decision and statement coverage) are used in experiments. Experiments and analysis of research of Li et al shows that Genetic algorithm performs well and greedy approaches are surprisingly effective [13].

Zhongsheng collected user session data from user session record that is generated on web server and proposed user session based test suite optimization approach based on genetic algorithms. User session is divided into meaningful user session using ReducedUSession algorithm. Genetic algorithm is applied on reduced user session using error coverage ratio and cost of test run as fitness function. This approach was applied on a small web login system. The results of experiments show that this approach is not good to find the additional faults in the system but it can reduce the test suite and save testing time [14].

Nachiyappan et al used multi-objective genetic algorithm for test suite minimization. Two objective functions coverage and execution time are selected. Selections of test cases were made by using roulette wheel selection algorithm. EMMA tool was used to find the block coverage. This algorithm was applied on three projects and reduced test suite was generated. However, this algorithm can be improved by using statement coverage and decision coverage to get better result [15].

Kaur et al prioritized regression test suites using genetic algorithm. Faults detection rate is maximized and time to detect faults is minimized in this prioritization approach. Genetic algorithm with proposed fitness functions was implemented in Java and test suites of five programs were evaluated, efficiency of 50 to 100% was obtained. The tool created for optimization purpose only accepts integer input and was not applied on commercial software [16].

In another study, Kaur et al used GA for regression test suite prioritization but total code coverage is used as fitness function. Proposed algorithm was applied on triangle problem and prioritized test suite was obtained which can be used to reduce cost and effort of testing [17].

2-4-2 RTO AND SWARM ALGORITHMS

Kaur et al prioritize regression test suites using Bee Colony algorithm. The algorithm consist of three step; path construction in that path is constructed by the scout bees and this path is saved in path of scout bees (PSB) table, in path restructuring forager bees exploit the PSB and uses PSB table to forms the forager bees (PFB) table. In PFB table test cases with best values are saved, in path selection phase path with minimum execution time is selected. This algorithm was compared with the various algorithms and proved that it gives better results [18].

Souza et al proposed binary multi-objective swarm particle optimization using crowd distance and Roulette wheel for automatic test suite generation. They compared binary multi-objective swarm particle algorithm and binary multi-objective swarm particle optimization. Best Pareto front for objective function requirement coverage & cost was used. BMOPSO-CDR approach outperformed BMOPSO and the Random method [19].

Kaur et al proposed Hybrid Particle Swarm Optimization for test suite prioritization that is combination of PSO and GA. To eliminate the chances of convergence to local minimum and create diverse population mutation operator is introduced into PSO. Percentage of faults detected is used as metrics; and best test suite is that covers maximum faults in minimum time. Algorithm was implemented in Java and 75.6% faults coverage is achieved but execution time of algorithm is quite long [20].

Another approach proposed by Kaur et al is PSO & GA based approach for test suite prioritization. In this approach, chance of convergence to local minima and to increase diversity among population crossover operator is introduced. Said approach was implemented in Java. This approach was applied on five programs and two metrics were used for analysis. Fault based experiment resulted APFD =90.3% in best case. Path based experiment resulted APCC 93.1% in best case [21].

3- OPTIMIZATION PROBLEM FORMULATION FOR RTO

RTO can be applied on White Box Testing and Black Box Testing with the change in parameters or objective functions [3]. In this research, authors considered Black Box based testing methods for the optimization of regression test suite.

3-1 RTO AND BLACK BOX TESTING

Functional testing of system is Black Box Testing. System is taken as black box and its functions are tested by inputting data and evaluating outputs of system. Regression test suite that is designed for Black Box Testing may use Requirement Change, Test Case Behavior in previous release, Architecture Changes, Failure Rate, Test execution time, Implementation Complexity, Defects Detection Rate, Configuration files changes, Database files changes, User Session, Cost, Customer Priority, Requirement Traceability, Fault Impact of Requirement, Data Access Cost, Technical Resources Cost, Setup Cost, Simulation Cost, Fault Model Sensitivity, Fault History Sensitivity, Human Sensitivity, Business Sensitivity, Precedence, Dependence Constraint, Conjunction Constraint, Exclusivity Constraint and Customer Priority of Requirement as objective functions [14], [22]-[29].

3-2 OPTIMIZATION OBJECTIVES

In this study, four objective functions are defined for the optimization of regression test suites, these objectives were measured during the original run of test suite and recorded into database. Optimization of regression test suite was made based on these objectives.

Test cases are written to find faults in programs; a test case can detect zero to many faults. Authors calculated the faults detected by every test case during the run of original test suite. The number of faults detected by a test case is faults detection rate and can be calculated by following formula:

Fault Detection Rate (F_{DR}) = Faults Detected / Total Faults (1)

Time taken to execute the test case is execution time. Authors calculated the execution time of each test case during the execution of original test suite by inserting timer in the code and test scripts.

In Black Box Testing, test cases are created to cover the requirements. Requirement Coverage measures the number of requirements covered by a test case. Authors calculated the requirement coverage of each test case during the test case engineering. Every requirement is divided into equivalent classes and test cases are written using equivalent class partitioning and boundary value analysis. Authors maintained the traceability of requirements and test cases. The requirement coverage of each test case is calculated by following formula:

Requirement Coverage (Rc) = Requirements Covered / Total Requirements (2)

Requirement Failure Impact is a reliability parameter. The analyst or test manager assigns it during the documentation of requirements. It defines the intensity of impact if a requirement fails or not properly implemented. Its value will be 0 to 1 based on the criticality of requirement. It makes the optimization process safe by introducing criticality of requirements during the optimization process.

3-3 PROBLEM FORMULATION

Regression test suite is optimized by minimizing the test suite to save the regression testing time in this study. In the literature [3], 27 objective functions are identified and used for the Black Box based RTO. Four objectives for optimization are used in this study; these objectives will be measure during the testing of software and recorded into test case history. These objectives are Fault Detection Rate, Execution Time, Requirement Coverage and Requirement Fault Impact. These objectives are selected after literature review and discussion with Software Testing experts. Following variables are required to represent the problem mathematically:

Table 1: Notations for Problem Formulation

Description	Notation
Test Case	T_C
Test Suite	T
Optimized Test Suite	T_o
Requirement Coverage	R_C
Requirement Failure Impact	R_{FI}
Fault Detection Rate	F_{DR}
Execution Time	E_T
Suitability of Test Case	S
Best Suitable	B_S
Moderate Suitable	M_S
Normal Suitable	N_S

Test Suite T is collection of n test cases; where n is total number of test cases in test suite.

$$T = \sum_{i=1}^{n} T_{ci} \quad (3)$$

Suitability of a test case to become part of optimized test suite is defined as High, Medium and Low and test suite belongs to only one class at a time.

$$S = N_S \cup M_S \cup B_S \quad (4)$$

Objective function for finding To is to save test suite execution time. It consists of multiple objective functions that are:

$$\text{Max } F_{DR}(T) \quad (5)$$
$$\text{Min } E_T(T) \quad (6)$$
$$\text{Max } Rc(T) \quad (7)$$
$$\text{Max } R_{FI}(T) \quad (8)$$

4- CASE STUDY 1: PREVIOUS DATE PROBLEM

Previous Date Problem was selected and test suite of Previous Date Problem was optimized using the said approaches for multi-objective optimization. Previous Date Problem is selected because it is a small problem and understanding of results is easy. Details of Previous date problem along-with code and test suite is available at [30]. 33 test cases were created using Equivalent Class Par-

titioning[3] and Boundary Value Analysis[4]. Faults Seeding[5] was used to insert six faults in the original program by interchanging the relational operators and increment / decrement operators. Authors executed the test suite and measured Faults Detection Rate, Execution Time, Requirement Coverage and Requirement Failure Impact of each test case of Previous Date Problem. Our approach of optimization is independent of process of measurement of these objectives because these objectives can be measured automatically and manually and approach is suited for both methods of measurement. Authors performed nine experiments to optimize regression test suite of Previous Date Problem with Multi-Objective Genetic Algorithms (MOGA), Non-Dominated Sorting Genetic Algorithm (NSGA-II) and Multi-Objective Particle Swarm Optimization (MOPSO); three experiments were performed on each approach considering the stochastic nature of these approaches.

Three experiments were performed to optimize the regression test suite using MOGA [31] and Matlab Global Optimization Toolbox [32] was used because MOGA solver is already implemented in Matlab. 93 generations were generated using 108 iterations with MOGA and generated optimized test suite. Same setting was followed in all experiments with MOGA. Suitability of our test cases lies between 0 to 1. Pareto Fronts generated by MOGA are shown in Figure 1. Three-dimensional Pareto is drawn and 4th dimension (RFI) is represented by color.

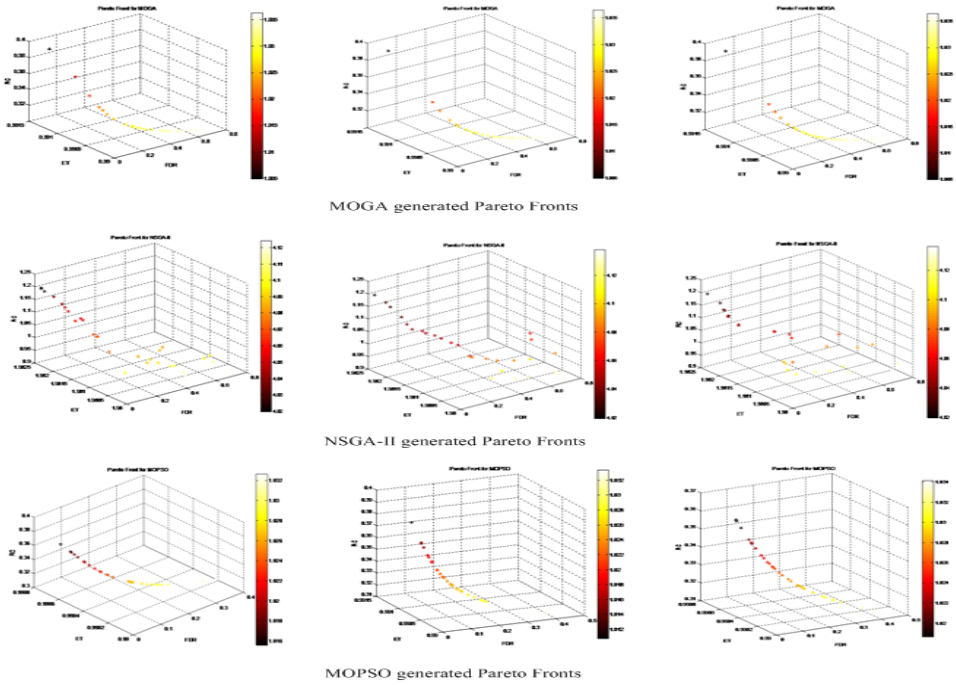

Figure 1: Pareto Front generated for Case Study 1

[3] Process of dividing input domain of program into equivalent classes.
[4] Input / Output value that is on the edge of equivalent classes.
[5] Faults' seeding is a process of inserting faults in programs, to check efficiency of test cases.

Three experiments were performed using Non Dominated Sorting Genetic Algorithm-II (NSGA-II) that is a multi objective Genetic Algorithm developed by Deb et al [33]. It is used in literature [34] for Regression Test Suite Optimization. The Matlab Code of NSGA-II written by Aravind Seshadri [5] was modified to optimize the regression test suite. 33 populations and 100 generations were created because original test suite consists of 33 test cases. Pareto Fronts generated by NSGA-II are shown in Figure 1.

Particle Swarm Optimization (PSO) [35] is very useful for RTO and is used in many studies [12,19,21]. Multi Objective Particle Swarm Optimization (MOPSO) [36] MOPSO code programmed by S. Mostapha Kalami Heris was used in these experiments. This code can be obtained by emailing Carlos A. CoelloCoello (ccoello@cs.cinvestav.mx). Luciano S. de Souza [19] used the same code for Test Suite Selection. The code was modified to optimize four objective problems and the three experiments were repeated for the optimization of regression test suite. Pareto Fronts of repository are shown in Figure 1.

5- CASE STUDY 2: PRINT TOKENS

Siemens Print Tokens [37] is a lexical analyzer; it consists of 539 lines of code 18 procedures, 7 faulty versions and having total of 7 faults and 4130 test cases. Test cases were created by using TSL Tool6. Print Token Code, Test Suite and Faulty versions are available at SIR7 . Authors used print tokens in Case Study 2. Linux (Fedora 15) platform was setup to perform controlled experiments as per guidelines [38]. Authors executed the test suite and measured previously defined objectives. Test Suite of Print Tokens was optimized using MOGA, NSGA-II and MOPSO, three experiments were performed using each approach, and nine experiments were performed. Pareto Fronts are shown in Figure 2.

[6] A complier in which we input specifications and functional characteristics of software with runtime testing environment and it generates executable test scripts.

[7] Software-artifact Infrastructure Library (http://sir.unl.edu/portal/index.php)

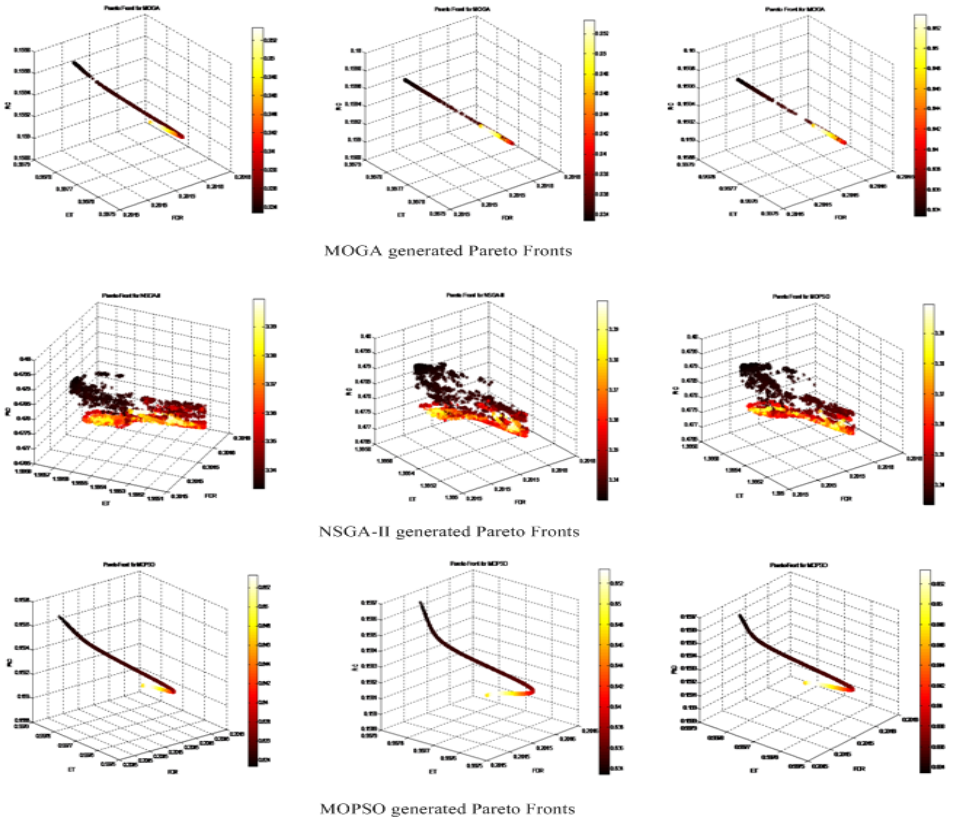

Figure 2: Pareto Fronts generated for Case Study 2

6- COMPARISON AND ANALYSIS

One can compare the Evolutionary Algorithms by comparing the generated Pareto Fronts shown in Figure 1 and Figure 2. By analyzing the Pareto Fronts, one can observe that fronts drawn by MOGA and MOPSO are better than the NSGA-II front. MOGA Pareto Front is better than the MOPSO Pareto Front because in MOGA Pareto Front ineffective solutions lie in a uniform region. For deep evaluation and comparison of both of these approaches, authors used TOPSIS [39] to get single result from the Evolutionary Algorithms using a VBA Tool Sanna. Authors defined four evaluation metrics to compare the results of Evolutionary Algorithms. Percentage Size Reduction of Test Suite is used to calculate the reduction in size of test suite after optimization. It can be calculated using following formula:

$$\text{Percentage of Test Suite Reduction} = \frac{|T| - |T'|}{|T|} X\ 100 \quad (9)$$

Where T is original Test Suite and T' is optimized Test Suite.
Percentage Loss in Faults Detection Rate can be determined by equation (10),

it measure that optimized test suite is safe or not. An unsafe optimization technique reduces the Faults Detection Rate of Test Suite.

$$\text{Percentage of Faults Detection Loss} = \frac{|F_{DR}| - |F_{DR}'|}{|F_{DR}|} X\ 100 \quad (10)$$

Where F_{DR} is Fault Detection Rate of original Test Suite and $F_{DR'}$ is Fault Detection Rate of Optimized Test suite.

Percentage Requirement Coverage Reduction is used to determine the reduction in requirement coverage of test suite after optimization. It can be calculated by using equation (11).

$$\text{Percentage of Requirement Coverage Loss} = \frac{|R_C| - |R_C'|}{|R_C|} X\ 100 \quad (11)$$

Where RC is Requirement Coverage of Original Test Suite and RC' is Requirement Coverage of Optimized Test Suite.

Percentage Time reduction is reduction in the execution time of test suite after optimization. It can be calculated by putting value of time in equation (9).

6-1 CASE STUDY 1: COMPARISON OF ALL TECHNIQUES

Comparison of all optimization techniques used in our experiments for Case Study 1 for our evaluation metrics shows that MOGA performs well as compared to NSGA-II and MOPSO. Evaluation results are shown in Figure 3.

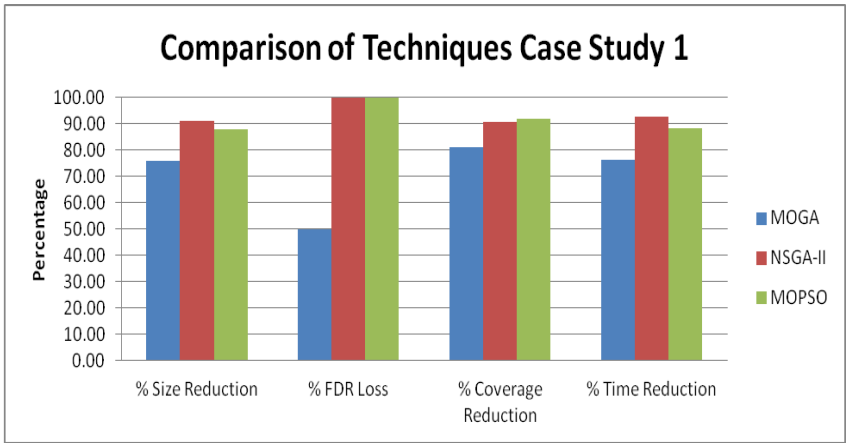

Figure 3: Case Study 1 Comparison and Analysis

6-2 CASE STUDY 2: COMPARISON OF ALL TECHNIQUES

Comparison of all optimization techniques used in our experiments for Case Study 2 for our evaluation metrics shows that MOGA performs well as compared

to NSGA-II and MOPSO for all our evaluation metrics. Evaluation results are shown in Figure 4.

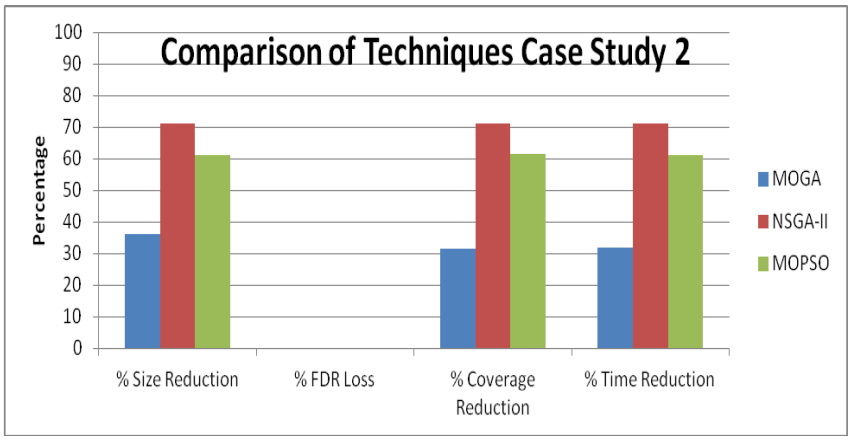

Figure 4: Case Study 2 Comparison and Analysis

7- CONCLUSION AND FUTURE WORK

Authors implemented and compared the three evolutionary algorithms for the optimization of regression test suite and experiments were performed on two published case studies. Pareto Fronts and previously stated four evaluation metrics were used for the comparison. After comparison and evaluation, one can conclude that MOGA is better for the optimization of regression test suite as compared to NSGA-II and MOPSO. Test suite optimized with MOGA resulted in minimum loss in faults detection rate and requirement coverage as compared to NSGA-II and MOPSO. Next candidate for the optimization of regression test suite is MOPSO. Authors also observed that MOGA is not good enough to reduce the size and execution time of optimized test suite. Faults detection rate of MOGA optimized test suite is also reduced for Case Study 1 therefore; MOGA is not a safe technique for the optimization of regression test suite. There is further need to explore other algorithms for RTO that have better faults detection rate and requirement coverage as compared to MOGA.

Authors are working to find better technique for RTO and our future work include solving the same case studies with Fuzzy Logic and ANFIS [40] because testing is a critical activity and expert judgment is required for selection of test cases. Fuzzy Logic [41,42] and ANFIS may be better candidates for regression test suite optimization and can make process of regression test suites optimization safe.

REFERENCES

[1] T. Takagi, and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," IEEE Transactions on Man and Cybernetics Systems, pp. 116-132, 1985.

[2] M. Kumar, A. Sharma and R. Kumar, "Optimization of Test Cases using Soft Computing Techniques: A Critical Review," WSEAS Transactions on Information Science and Applications, Issue 11, Vol. 8, pp. 440 – 452, November 2011.

[3] Z. Anwar, A. Ahsan, "Exploration and Analysis of Regression Test Suite Optimization," ACM SIGSOFT Software Engineering Notes, Volume 39 Issue 1, pp. 1-5, Jan. 2014.

[4] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," Swarm and Evolutionary Computation, Vol. 1, Issue 1, pp. 32-49, ISSN 2210-6502, http://dx.doi.org/10.1016/j.swevo.2011.03.001, March 2011.

[5] A. Pratap, S. Agarwal, T. Meyarivan, "A Fast Elitist Multiobjective Genetic Algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, Vol. 6 , Issue 2, pp. 182 – 197, April 2002.

[6] W. E. Wong, J. R. Horgan, S. London and H. Agrawal, "A Study of Effective Regression Testing in Practice," 8th IEEE International Symposium on Software Reliability Engineering (ISSRE'97), Albuquerque,NM, pp. 264-274, November 1997.

[7] R. Malhotra, A. Kaur and Y. Singh, "A Regression Test Selection and Prioritization Technique," Journal of Information Processing Systems, Vol.6, No.2, pp. 235-252, June 2010.

[8] S. Prasad , M. Jain, S. Singh and C. Patvardhan, "Regression Optimizer A Multi Coverage Criteria Test Suite Minimization Technique," International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868, Foundation of Computer Science FCS, New York, USA Vol. 1– No.8, pp. 5-11, April 2012.

[9] G. Duggal, B. Suri, "Understanding Regression Testing Techniques," http://www.rimtengg.com/coit2008/proceedings/SW15.pdf, pp. 1-6, 2008.

[10] T. L. Graves, M. J. Harrold, J. Kim, A. Porter and G. Rothermel, "An Empirical Study of Regression Test Selection Techniques," ACM Transactions on Software Engineering and Methodology, Vol. 10, No. 2, pp. 184–208, April 2001.

[11] S. Sumathi and Surekha P., "Computational Intelligence Paradigms Theory and Applications using MATLAB," CRC Press Taylor & Francis Group, Boca Raton, London, Newyork, ISBN: 978-1-4398-0902-0, 2010.

[12] A. Kaur and D. Bhatt, "Hybrid Particle Swarm Optimization for Regression Testing," International Journal on Computer Science and Engineering (IJCSE), Vol. 3, No. 5, pp. 1815-1824, May 2011.

[13] Z. Li, M. Harman, and R. M. Hierons, "Search Algorithms for Regression Test Case Prioritization," IEEE Transactions on Software Engineering, Vol. 33, No. 4, April 2007

[14] Q. Zhongsheng, “Test Case Generation and Optimization for User Session-based Web Application Testing,” Journal of Computers, Vol. 5, No. 11, pp. 1655-1662, November 2010.

[15] S. N. A. Vimaladevi, and C.B. S. Lakshmi, “An Evolutionary Algorithm for Regression Test Suite Reduction,” Proceedings of the International Conference on Communication and Computational Intelligence, pp. 503-508, 2010.

[16] A. Kaur, and S. Goyal, “A Genetic Algorithm for Fault based Regression Test Case Prioritization,” International Journal of Computer Applications (0975 – 8887) Vol. 32, No.8, pp 30-37, October 2011.

[17] A. Kaur and S. Goyal, “A Genetic Algorithm for Regression Test Case Prioritization Using Code Coverage,” International Journal on Computer Science and Engineering (IJCSE), ISSN: 0975-3397 Vol. 3, No. 5 pp. 1839-1847, May 2011.

[18] A. Kaur, “A Bee Colony Optimization Algorithm For Code Coverage Test Suite Prioritization,” International Journal of Engineering Science and Technology (IJEST), Vol. 3, No. 4 pp. 2786-2795, April 2011.

[19] L. S. de Souza, P. B. C. de Miranda, R. B. C. Prudencio and F. de A. Barros, “A Multi-Objective Particle Swarm Optimization for Test Case Selection Based on Functional Requirements Coverage and Execution Effort,” 23rd IEEE International Conference on Tools with Artificial Intelligence, pp. 245-252, 2011.

[20] S. Yoo and M. Harman, “Pareto Efficient MultiObjective Test Case Selection”.

[21] A. Kaur and D. Bhatt, “Particle Swarm Optimization with Cross-Over Operator for Prioritization in Regression Testing,” International Journal of Computer Applications (0975 – 8887), Vol. 27, No.10, pp. 27-34, August 2011.

[22] Nanda, Agastya, S. Mani, S. Sinha, M. J. Harrold, and A. Orso. "Regression testing in the presence of non-code changes," IEEE 4th International Conference on Software Testing, Verification and Validation (ICST), pp. 21-30, 2011.

[23] S. Parsa and A. Khalilian, “On the Optimization Approach towards Test Suite Minimization”, International Journal of Software Engineering and Its Applications Vol. 4, No. 1, pp. 15-28, January 2010.

[24] G. Whyte, and D. L. Mulder, “Mitigating the Impact of Software Test Constraints on Software Testing Effectiveness,” The Electronic Journal Information Systems Evaluation Vol. 14, Issue 2, pp. 254-270, 2011.

[25] Xu, Zhiwei, KehanGao, and T. M. Khoshgoftaar. "Application of fuzzy expert system in test case selection for system regression test," IEEE International Conference on Information Reuse and Integration (IRI), 2005.

[26] E. Ashraf, A. Rauf, and K. Mahmood. "Value based Regression Test Case Prioritization," Proceedings of the World Congress on Engineering and Computer Science, Vol. 1, 2012.

[27] M. Harman, "Making the case for MORTO: Multi objective regression test optimization," IEEE 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2011.

[28] S. Raju, and G. V. Uma. "Factors oriented test case prioritization technique in regression testing using genetic algorithm," European Journal of Scientific Research 74.3, pp. 389-402, 2012.

[29] S. Yoo and M. Harman, “Regression Testing Minimisation, Selection and Prioritisation: A Survey," Software Testing Verification and Reliabiability, pp. 1-60, 2007.

[30] K.K. Aggarwal, and Y. Singh, “A book on software engineering,” New Age International (P) Ltd.; Publishers, 4835/24, Ansari Road, Daryaganj, New Delhi, 2001.

[31] Konak, Abdullah, D. W. Coit, and A. E. Smith. "Multi-objective optimization using genetic algorithms: A tutorial," Reliability Engineering & System Safety, pp 992-1007, 2006.

[32] Branch, M. Ann, and A. Grace. “MATLAB: optimization toolbox: user's guide,” version 1.5. The MathWorks, 1996.

[33] K. Deb, S. Agrawal, A. Pratab, T. Meyarivan, “A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II,” Proceedings of the Parallel Problem Solving from Nature VI Conference, Springer. Lecture Notes in Computer Science No. 1917, Paris, France, pp. 849-858, 2000.

[34] Yoo, Shin, and M. Harman. "Using hybrid algorithm for pareto efficient multi-objective test suite minimization," Journal of Systems and Software, pp. 689-701, 2010.

[35] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 1942–1948, 1995.

[36] Coello, C. A. Coello, G. Toscano Pulido, and M. S. Lechuga. "Handling multiple objectives with particle swarm optimization," IEEE Transactions on Evolutionary Computation, pp. 256-279, 2004.

[37] Hutchins, Monica, et al. "Experiments of the effectiveness of dataflow-and control flow-based test adequacy criteria," Proceedings of the 16th international conference on Software engineering. IEEE Computer Society Press, 1994.

[38] Do, Hyunsook, S. Elbaum, and G. Rothermel. "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," Empirical Software Engineering, pp. 405-435, 2005.

[39] Jablonský, Josef, "Software Support for Multiple Criteria Decision Making Problems," Management, pp. 029-034, 2009.

[40] J.Roger Jang, C. Sun and E. Mizutani, “Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Learning,” Prentice Hall Upper Saddle River, NJ 07458, ISBN: 0-13-261066-3, 1997.

[41] E. H. Mamdani, S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller,” International Journal of Man-Machine Studies, pp. 1–13, 1975.

[42] Z. Anwar, “Neuro-Fuzzy Modeling based Regression Test Suite Optimization,” M.Sc Discretion, Center for Advance Studies in Engineering, Islamabad, Pakistan, 2013.