Analysis of MORTO:Multi-objective Regression test optimization

Neha Gupta[1], Arun Sharma[2], Manoj Kumar Pachariya[3]
[1,2]Indira Gandhi Delhi Technical University for Women New-Delhi, India, [3]MCNUJC, Bhopal, India

***Abstract*—Whenever a change is done in software, regression testing is done to check that a recent change in code has not created any unwanted defects in the system. As size of regression test suite is very large, optimization algorithms help in selecting, minimizing and prioritizing test suites. Main aim is maximize fault detection ability with less number of test cases. Optimization techniques of various types are available but Multi-objective algorithms are the best choice to use as testing is dependent on many adequacy criteria or surrogates. In this paper, authors have carried out study on research papers where multi-objective algorithms are used in regression testing. Then comparison of performance of multi-objective algorithms is done to identify the best suitable multi-objective algorithm for regression testing.**

Keywords—Regression testing; Multi-objective algorithms; optimization techniques

I. Introduction

Regression testing is a testing technique which is conducted to approve that a recent change in code has not created unwanted defects in the changed system. Regression testing is generally done when there is a change in code due to requirement change; a new feature is added to the software; a defect is corrected; or code is fixed for performance issues. To retest that the existing functionalities of software work fine, one approach is to implement all the available test cases in the test suite. This is very expensive as mostly software code and there test cases are very large in number. Thus, executing all of them requires lot of time and resources.

In order to lessen the time and resources requisite for regression testing, various approaches to select minimize and prioritize test cases have been developed. When few test cases are selected only to test the modified parts of code then it is called test case selection. Removing redundant or similar test from test suite for retesting the software is called test suite minimization. Prioritizing test cases means ordering test case execution such that the desired properties such as frequently used functionalities, critical requirements or early fault detection should be covered.

Another important part in software testing is deciding type of adequacy criterion that should be used for optimizing test cases. Adequacy criteria are the regulations used for testing to verify that all software errors have been identified and removed. Different types of adequacy criteria are there such as control flow (branch, path, loop), data flow (de & all-use), and fault-based (K score, D score) adequacy criteria. These adequacy criteria act as a surrogate on basis of which regression testing may also be carried out.

Depending on one surrogate or adequacy criteria will give uncertain results. Thus it's better to consider many criteria together and optimize test cases on basis of them. Various researchers such as Mondal et al. [5], Panichella et al. [7],Turner et al. [14],Yoo et al. [15] etc have optimized test cases based on multiple criteria.

For optimization various techniques may be used but as regression testing is a multi-criteria problem. Technique should be selected such that it helps for multi-criteria optimization. For solving a multi-criteria problem, either you can aggregate many objectives into one objective function by weighted sum approach and find a solution or use a multi-objective algorithm which provides solution in form of Pareto fronts. Murata and ishibuchi [1] stated that "if constant weights are associated with objectives, then the search direction is fixed and if multi-objective optimization problems have concave Pareto fronts, then weighted sum approaches tend to fail to find entire Pareto fronts". Thus it's better to use multi-objective algorithms rather than weighted sum approach. With new advancements in multi-objective algorithms there is scope of improvement in optimization of regression testing process. Gupta et al. [2] have reviewed optimization work in software testing and identified adequacy criteria and algorithms which are rarely used. But they did not compare the results of existing optimization algorithms. In this research work, we have conducted a study on research work where multi-objective algorithms have been used for regression testing. Our main aim is to identify the best available multi-objective algorithms which have been used for regression testing and shown good results in most of the cases. So, that it can be used as a benchmark algorithm in future studies also. For conducting this analysis, time period of 2010-present has been taken and multi-criteria optimization works in regression testing where multi-objective algorithms have been used are considered.

Research tasks completed in the study are:

RT 1: To find research work where multi-objective optimization algorithms have been used in various fields of regression testing.

RT 2: Finding the multi-objective algorithm which gives best result for various fields of regression testing.

Further the paper comprises of given sections: Section 2 consists of the literature review in all fields of regression testing i.e. selecting, minimizing and prioritizing test cases.

Section 3 consists of the analysis of considered literature. Conclusion is in Section 4.

II. Literature Review

In this section, research work which is either selecting, minimizing or prioritizing test cases using multi-objective algorithms have been reviewed.

A. *Multi-Objective Algorithms in Test case selection*

De Lucia et al. [3] have increased diversity in the solutions provided by NSGA-II and used code coverage and execution time for selecting test cases. Results after evaluation on programs from Software Infrastructure Repository show that convergence speed of enhanced NSGA-II is better compared to other considered algorithms and it provides more optimal diversified solutions.

A combined algorithm of binary PSO and MO-PSO with crowding distance approach has been proposed by De Souza et al. [4] for selecting test cases. Five different versions of proposed algorithm have been used to optimize requirement coverage and cost. Results were validated on integration and regression test suites from mobile devices background. Of all five versions, Novel BMOPSO-CDR outperformed others for both considered test suite i.e. Integration and Regression in respect to Generational distance and Coverage.

Mondal et al. [5] have used NSGA-II algorithm to optimize test diversity and code coverage together for selecting test cases. They optimized both coverage with execution time using NSGA-II and Additional greedy algorithm. Results show that with three objectives to optimize NSGA-II gave better results than Additional greedy algorithm.

BMOPSO-CDR and BMOPSO-CDRHS algorithms proposed by De Souza et al. [6] are better then compared to NSGA-II and MBHS. Results show that BMOPSO-CDRHS performed the best.

Diversity based Multi-objective GA has been proposed by Panichella et al. [7] for selecting test cases on the basis of coverage, past fault coverage and execution time. Results on 11 projects from SIR repository show that it performs best compared to NSGA-II and Additional greedy algorithms.

Pradhan et al. [8] have used cost and test effectiveness for selecting test cases. SPEA2 and Random Search have been used for optimization and SPEA2 performed the best.

Choudhry et al. [9] have used Multi-objective Harmony search algorithm for selecting test cases on basis of fault coverage and execution time. MO-Harmony search algorithm gave 100% fault detection compared to bat and cuckoo search.

Saber et al. [10] have proposed GREAP algorithm based on greedy, genetic and local search algorithm. Greedy quickly finds superior solutions, genetic algorithm increases the covered search space and local search help in refining the solutions. GREAP outdid NSGA-II and MOEA/D for statement, branch and MC/DC coverage objectives.

B. *Multi-Objective Algorithms in Test case Minimization*

Yoo and Harman [11] have proposed a hybrid of MOGA and greedy approach for selecting test cases based on coverage and past fault detection. Proposed algorithm was able to produce solutions which were able to dominate the solutions of additional greedy algorithm.

Multi-Objective TAEA has been used by Yoo et al. [12] for optimizing test cases on basis of statement coverage, execution time, and fault history for a Google project. Due to fault history coverage a reduce test suite with fault detecting test cases in them was produced as a result.

Kumari et al. [13] have proposed Quantum based MO-DE Algorithm for minimizing test suites on the basis of code coverage and time for execution. Its comparison has been done with NSGA-II algorithm. QMDEA has faster convergence rate than NSGA-II.

A test suite reduction strategy for Mockito project is developed by Turner et al. [14]. They have optimized test suite on the basis of branch coverage and execution time with help of NSGA-II algorithm. Results show that with reduced test suite there is 50% reduction in execution time with 96% branch coverage.

Execution time of multi-objective algorithms on general CPU is very high. GPGPU has feature of parallelism which helps in improving scalability of SBSE. Yoo et al. [15] have implemented NSGA-II, SPEA2 and TAEA on GPGPU for test suite minimization. Results show that there is increase in execution speed of 25.09 times compared to normal execution.

Optimization techniques: Greedy, NSGA-II and MOEA/D have been used by Zheng et al. [16] for test suite minimization. Results demonstrate that MOEA/D is the strongest algorithm compared to other considered algorithms. Both NSGA-II and MOEA/D were competitive enough.

Wei et al. [19] have minimized test suite based on mutation score, code coverage and cost. For optimization they have used NSGA-II, IBEA, MOEA/D variants and SPEA-II. Result depicts that NSGA-II and MOEA/D are more competent and better than IBEA and SPEA-II algorithms.

C. *Multi-Objective Algorithms in Test case Prioritization*

Processing on normal CPU takes time. Li et al. [20] have used GPU-based parallel MOEAs to reduce the execution time required for prioritizing test cases. Study on eight project show there is increase in speed of around 120 times.

A prioritization technique based on code coverage and changed code coverage adequacy criteria has been proposed by Epitropakis et al. [21]. They have used Additional greedy algorithm, NSGA-II, and TAEA for optimization. Execution time and coverage data for considered algorithms has been reduced due to compaction algorithm.

NSGA-II algorithm based tool called MOTCP+ has been proposed by Marchetto et al. [22] for maximizing fault detection by optimizing cost, code, and requirement coverage. APFD for MOTCP+ is 80% whereas for AddCodeCov and NSGAIIdim2 is 66.6% and 62% respectively.

A hyper volume based evaluation for prioritizing test cases has been done by Di Nucci et al. [23]. Results show that HGA outdid Additional greedy algorithm for four of the case studies.

Wang et al. [24] have developed a resource aware test prioritization technique i.e. it is aware with the resource usage and time required. GA and its variants, SPEA-II, PSO and hybrid of GA-DE have been used for optimization. Among all random weighted genetic algorithms performs the best.

Parejo et al. [25] have used NSGA-II algorithm fro test case prioritization in highly configurable systems. Various functional and non functional criteria have been used on the basis of which optimization is done. Results show better prioritization result by NSGA-II compared to random technique.

A test case prioritization technique for web application has been proposed by Khanna et al. [28]. They have used NSGA-II algorithm for optimizing tests based on fault detection and time required for execution. Performance of NSGA-II has been compared with weighted GA as well as with greedy and its variants. Results show that NSGA-II performs the best.

Multi-objective genetic algorithm has been used by Mishra et al. [30] for test case prioritization. Factors considered for prioritizing test cases are statement and requirements coverage, risk exposure, and execution time. Results show that MOGA was able to effectively prioritize test cases with early fault detection.

III. Analysis

A. *Research Task 1:*

Analysis of multi-objective algorithms used for selecting, minimizing and prioritizing test cases is given in Table 1, Table 2 and Table 3 respectfully. With this our first task of gathering literature and presenting it is complete.

B. *Research Task 2:*

Second task is to find MO-algorithm which can be used as benchmark algorithm to solve regression testing problem. To find out benchmark algorithm, performance of multi-objective algorithm need to be checked and performance is dependent on various factors such as objective functions which need to be optimized, setting of parameters for considered algorithm, subject under study etc. We tried to gather all this information from each research work so that comparison of multi-objective algorithms can be done. We also noted if any comparison is done with other algorithm in considered research work. To compare two work of optimization, the subject under test (SUT) should be same. So, we have also compared the results of two different research works, if they are tested on same SUT.

From this study it can be seen that multi-objective algorithms such as NSGA-II, MOPSO, SPEA-II, MOEA/D, MO-Harmony search, two archive MO algorithm, MO-GA, MO-ACO are widely used by researchers. From Fig. 1, it may be observed that NSGA-II is the most used and validated algorithm. In most of the cases it performed better than other considered algorithms but few cases are there such as [6], [10], [13], [16] where it underperformed. In these cases BMOPSO-CDRHS, GREAP, QMDEA, MOEA/D performed better than NSGA-II. Cases where NSGA-II performs better are [5], [19], [21], [25], [28]. In these cases NSGA-II outdid Additional greedy, SPEA-II, IBEA, MOEA/D, random optimization, MO-GA. In most cases where NSGA-II, MOEA/D and SPEA-II algorithms are used for same problem, NSGA-II and MOEA/D outdid SPEA-II. After NSGA-II mostly used algorithms are variants of MO-GA and they mostly outdid Additional greedy algorithm.

In prioritization it can be seen that MOTCP technique is considered in [21], [22] and [26]. In these cases, NSGA-II and TAEA performed better then additional greedy technique. Epistatis ACO outperformed ACO and NSGA-II. Thus for MOTCP it can be said that epistasis ACO is better so far.

From all these observation we came to conclusion that most studied MO-algorithm is NSGA-II, and then MOEA/D and SPEA-II are studied. Among these three NSGA-II and MOEA/D are better compared to SPEA-II in most cases.

As a benchmark MO-Algorithm we consider NSGA-II to take this position as most of the researchers consider it as an algorithm and in mostly cases it has outdid or performed equivalent to other considered algorithm. But it’s not necessary that it will give good results always, as already discussed that BMOPSO-CDRHS, GREAP, QMDEA, MOEA/D algorithms outdid NSGA-II in few cases. This can be due to factors such as type of applications, considered functions for optimization and parameters set.

Thus, researchers may try to use NSGA-II as the base algorithm.

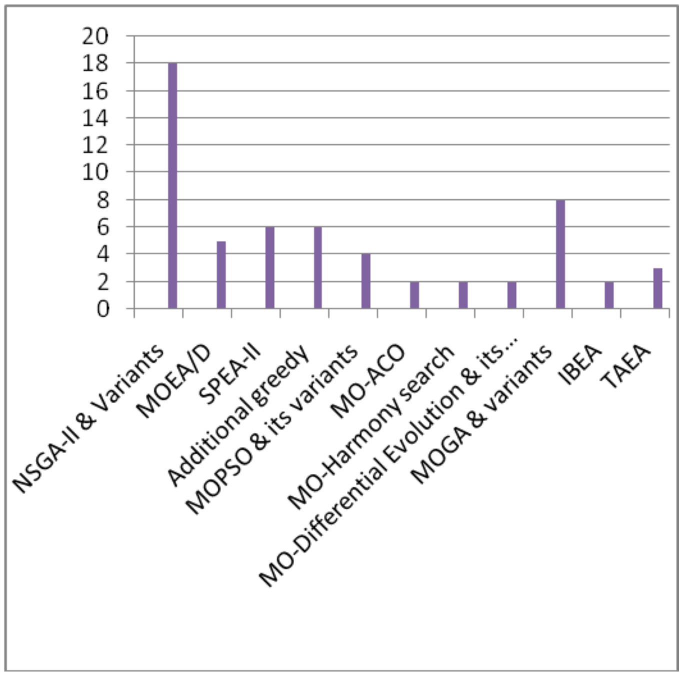

Fig. 1. No. of publication in which considered multi-objective algorithm is used.

TABLE 1. MULTI-OBJECTIVE ALGORITHMS USED IN TEST CASE SELECTION

Paper	Multi-objective Algorithm	SUT	Criteria	Value of parameters	Comparison and Results
[3]	NSGA-II with Diversity	printtokens, printtokens2, schedule and schedule2	Code coverage Execution cost	Population Size: 300 No of iterations: 300 No. of partitions: 20	Compared with basic NSGA-II and gave better results in terms of optimal solution and convergence rate.
[4]	BMOPSO-CDR	Mobile system test suite (Integration and regression test suites)	Requirement coverage and cost	No. of particles: 20 Mutation rate: 0.5 constant C1 &C2: 1.49 Iterations: 200,000	NBMOPSO-CDR out-performed other versions and Binary PSO
[5]	NSGA-II	Apache Ant, Derby, Jboss, NanoXML, Math	Code coverage Test diversity Test execution time	Population size: 200 No. of iterations: 5000	Compared with Additional greedy algorithm. NSGA-II performs better.
[6]	BMOPSO-CDR and BMOPSO-CDRHS	Flex, Grep, Gzip, sed, Space, Mobile-system	Branch coverage Functional coverage	Iterations: 200,000 particles 20mutation rate:0.5, constants C 1 and C 2=1.49, Velocity=4.0	Compared with NSGA-II, Multi-objective binary harmony search and BMOPSO-CDR. BMOPSO-CDRHS performs the best among all.
[7]	DIV-GA (Diversity based genetic algorithm	Bash, flex, grep, gzip, printtoken, printtokens2, schedule, schedule2, sed, space, vim	Code coverage execution cost Past fault coverage	No. of iterations and population depends on project scattered crossover function with probability= 0:50	DIV-GA outdid both greedy and multi-objective GA
[8]	SPEA-2	Subsea Oil and Gas Production System and artificially created problem	Time Difference Mean Priority, Probability and Consequence	Not given	SPEA 2 outdoes Alternate variable method and random search in terms of mean priority, probability and consequences
[9]	Multi-objective Harmony search algorithm	Flex, Grep, Gzip, nano-xml, xml-security	Fault coverage, Execution time	No. of iterations:500 No. of particles 15	Compared with Bat and cuckoo algorithm. Results show that MO-Harmony search was able to find 100 % faults with even 5 test cases.
[10]	GREAP	Gzip, space, Schedule, Totinfo, Tcas, grep, Sed, Make	Statement, branch and MC/DC coverage.	Population Size: 100	GREAP gives better results than NSGA-II and MOEA/D

TABLE 2. MULTI-OBJECTIVE ALGORITHMS USED IN TEST CASE MINIMIZATION

Paper	Multi-objective Algorithm	SUT	Criteria	Value of parameters	Comparison and Results
[11]	Hybrid MOGA (MOGA +Greedy)	Flex, grep, gzip, sed, space	Coverage Past fault detection	No. of iteration 20, Cross-over and mutation rate=1	Hybrid algorithm was able to produce solutions which dominated solution by additional greedy algorithm.
[12]	Two-Archive Multi-Objective Evolutionary Algorithm	Google software	Code coverage Execution time Fault history	Not given	No comparison done. Reduced test suite was able to include fault revealing tests due to fault history.
[13]	QMDEA	Real world Application 1 Application 2	Code coverage Execution time	Population size: 100 No. of iteration: 10,000 Control parameter=0.8	Compared with NSGA-II, QMDEA has faster convergence rate and provided large number of solutions.
[14]	NSGA-II	Mockito	Branch coverage and execution time	Not given	Comparison not done with other algorithm. Reduced test suite provided 96 % coverage with 50% less execution time.
[15]	GPGPU+ (NSGA-II SPEA Two Archive algorithm)	Printtokens, tcas, printtokens2, schedule, schedule2, totinfo, replace, space, flex, gzip, sed, bash, haifa	Requirement coverage Computation cost	Population size : 256 number of threads :multiples of 32 or 64 No. of iteration:250	A speedup of 25.09X has been achieved by GPGPU based technique compared to CPU executed algorithms
[16]	Greedy technique NSGA-II MOEA/D	Gzip, Schedule, tcas, tot_info, voidAuth, space	Cost Statement coverage Branch coverage MC/DC coverage	Population size=120, Evaluations=100,000 Constant c is assigned fixed value of 0.3.	Results show that greedy technique is worst in performance. MOEA/D with fixed constant performed better than NSGA-II.
[17]	NSGA-II	Cyber physical system	Code coverage Execution time Uncertainty wise tests	Population size: 100, Selection operator: binary tournament	No comparison done. Blend Alpha Crossover operator helped NSGA-II to get best results.
[18]	NSGA-II	Terp Paint	Weight of test cases	Not given	No comparison done. Reduction of

		Notepad	No. of faults identified		20% in test suite size with compromise of15.6% in fault revealing capability.
[19]	NSGA-II MOEA/D and its variants IBEA SPEA-II	Tcas, Tot_info, Schedule, space	Mutation score, cost, statement, branch and MC/DC coverage.	Population: 100 Evaluation: 80000	In terms of hyper volume NSGA-II and MOEA/D variants and are efficient compared to IBEA and SPEA2.

TABLE 3. MULTI-OBJECTIVE ALGORITHMS USED IN TEST CASE PRIORITIZATION

Paper	Multi-objective Algorithm	SUT	Criteria	Value of parameters	Comparison and Results
[20]	GPU NSGA-II	Printtokens, printtokens2, schedule, schedule2, replace, tcas, flex, space, V8	Statement Coverage Execution time	Population size : 256 No. of iteration:250	Speedup of up to 120 times is achieved by using GPU.
[21]	NSGA-II, TAEA, Additional greedy algorithm	Flex, grep, gzip, make, sed, mySQL	Average % of coverage achieved, % coverage of changed code, % of past fault coverage.	Population size: 250, crossover rate: 0.9	NSGA-II and TAEA outperformed Additional greedy algorithm.
[22]	NSGA-II	LaTazza, AveCalc, Commons(Proxy, IO, BeanUtils, Lang), DBUtils, iTrust, Jtidy, Woden, Log4J, Betwixt, JXPath, XMLGraphics, Pmd, Jabref	Code coverage, Additional code coverage, Requirement coverage and cost	Population size=2*test suite size Iterations=1,000 Mutation Probability=1/test suite size	No comparison with other multi-objective algorithm done.
[23]	Hyper volume based GA	Bash, flex, grep, print_tokens, print_tokens 2, sed	Code coverage, Past fault coverage, Execution cost	Not mentioned	Hyper volume based GA performs better than Additional greedy algorithm.
[24]	NSGA-II, SPEA-II, PSO GA-DE hybrid	Cisco-video conferencing system	Time for execution, Resource usage and fault detection	Different parameters for each algorithm	Random weighted GA performs best among considered algorithms.
[25]	NSGA-II	Mobile phone system	Functional and Non functional criteria	Population size: 100 No. of iterations : 50 Crossover probability: 0.9	NSGA-II performs better than random optimization technique
[26]	Epistasis based ACO	Flex, space, bash, V8	Code coverage Requirement coverage	Population Size=32 For ACO; α = 1, β = 5, ρ = 0.1	It gives better result for MoTCP compared to general ACO and NSGA-II for V8 project
[27]	NSGA-II, MOCell, SPEA2 and CellDE, and Random Search (RS),	2 Cyber physical case studies	Average uncertainty measure, cost,	Default setting for each algorithm	Mocell then NSGA works best in all considered situations.
[28]	NSGA-II	5 Web applications	Fault detection Execution cost	Default settings	NSGA-II performs better than other considered techniques.
[29]	RIPPER NSGA-II, SPEA2 IBEA	Cisco Data Set1 Cisco Data Set2 ABB Paint Control ABB IOF/ROL Google GSDTSR	Fault detection capability, Test reliance score Execution time	Population: 100, Parents selection: Binary tournament, crossover rate = 0.9	IBEA performed best among all algorithm
[30]	MOGA	Income tax calculator	Code and requirement coverage, fault coverage, execution time	Default settings	No comparison

IV. CONCLUSION

In this research work authors have reviewed reearch works on multi-objective optimization in regression testing. This review is done for time period of 2010-present. Authors were able to find research work for fields of regression testing. From review it could be concluded that NSGA-II is the most used algorithm and can be used as benchmark algorithm as it outperforms many algorithms such as MOGA, Additional greedy algorithm, SPEA-II and MOEA/D for most of the regression testing optimization problems.

REFERENCES

[1] T. Murata, H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," In IEEE international conference on evolutionary computation 1995 Nov 29 (Vol. 1, pp. 289-294)

[2] N. Gupta, A. Sharma, MK Pachariya, "An Insight Into Test Case Optimization: Ideas and Trends With Future Perspectives" IEEE Access. 2019;7:22310-27.

[3] A, De Lucia, M. Di Penta, R. Oliveto, A. Panichella, “On the role of diversity measures for multi-objective test case selection,” In Proceedings of the 7th International Workshop on Automation of Software Test 2012 Jun 2 (pp. 145-151). IEEE Press.

[4] LS De Souza, RB Prudêncio, FD Barros, “A comparison study of binary multi-objective particle swarm optimization approaches for test case selection,” In 2014 IEEE Congress on Evolutionary Computation (CEC) 2014 Jul 6 (pp. 2164-2171). IEEE.

[5] D. Mondal, H. Hemmati, S. Durocher, “Exploring test suite diversification and code coverage in multi-objective test case selection,” In 2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST) 2015 Apr 13 (pp. 1-10). IEEE.

[6] LS de Souza, RB. Prudêncio, FA. de Barros, “A hybrid particle swarm optimization and harmony search algorithm approach for multi-objective test case selection. Journal of the Brazilian Computer Society,” 2015 Dec;21(1):19.

[7] A. Panichella, R. Oliveto, M. Di Penta, A. De Lucia, “Improving multi-objective test case selection by injecting diversity in genetic algorithms,” IEEE Transactions on Software Engineering. 2014 Oct 27;41(4):358-83.

[8] D. Pradhan, S. Wang, S. Ali, T. Yue, “Search-based cost-effective test case selection within a time budget: An empirical study,” In Proceedings of the Genetic and Evolutionary Computation Conference 2016 2016 Jul 20 (pp. 1085-1092). ACM.

[9] A. Choudhary, AP Agrawal, A. Kaur, “An effective approach for regression test case selection using pareto based multi-objective harmony search,” In Proceedings of the 11th International Workshop on Search-Based Software Testing 2018 May 28 (pp. 13-20). ACM.

[10] T. Saber, F Delavernhe, M Papadakis, M O'Neill, Ventresque A, “A hybrid algorithm for multi-objective test case selection,” In 2018 IEEE Congress on Evolutionary Computation (CEC) 2018 Jul 8 (pp. 1-8). IEEE.

[11] S. Yoo, M. Harman, “Using hybrid algorithm for pareto efficient multi-objective test suite minimisation,” Journal of Systems and Software. 2010 Apr 1;83(4):689-701.

[12]S. Yoo, R Nilsson, M Harman, “Faster fault finding at Google using multi objective regression test optimisation,” In 8th European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE’11), Szeged, Hungary 2011 Sep 5.

[13] AC Kumari, K Srinivas, MP Gupta, “Multi-objective test suite minimisation using quantum-inspired multi-objective differential evolution algorithm,” In 2012 IEEE International Conference on Computational Intelligence and Computing Research 2012 Dec 18 (pp. 1-7). IEEE.

[14] AJ Turner, DR White, JH Drake, “Multi-objective regression test suite minimisation for mockito,” In International Symposium on Search Based Software Engineering 2016 Oct 8 (pp. 244-249). Springer, Cham.

[15] S. Yoo, M. Harman, Ur S, “GPGPU test suite minimisation: search based software engineering performance improvement using graphics cards,” Empirical Software Engineering. 2013 Jun 1;18(3):550-93.

[16] W. Zheng, RM. Hierons, Li M, Liu X, Vinciotti V, “Multi-objective optimisation for regression testing,” Information Sciences. 2016 Mar 20;334:1-6.

[17] S. Ali, Y. Li, Yue T, Zhang M, “An empirical evaluation of mutation and crossover operators for multi-objective uncertainty-wise test minimization,” In 2017 IEEE/ACM 10th International Workshop on Search-Based Software Testing (SBST) 2017 May 22 (pp. 21-27). IEEE.

[18] N. Chaudhary, OP Sangwan, “Multi Objective Test Suite Reduction for GUI Based Software Using NSGA-II,” IJ Information Technology and Computer Science. 2016;8(8):59-65.

[19] Z. Wei, W. Xiaoxue, Xibing Y, Shichao C, Wenxin L, Jun L, “Test suite minimization with mutation testing-based many-objective evolutionary optimization,” In 2017 International Conference on Software Analysis, Testing and Evolution (SATE) 2017 Nov 3 (pp. 30-36). IEEE.

[20] Z. Li, Y Bian, R Zhao, Cheng J, “A fine-grained parallel multi-objective test case prioritization on gpu,” In International Symposium on Search Based Software Engineering 2013 Aug 24 (pp. 111-125). Springer, Berlin, Heidelberg.

[21] Epitropakis MG, Yoo S, Harman M, Burke EK, “Empirical evaluation of pareto efficient multi-objective regression test case prioritisation,” In Proceedings of the 2015 International Symposium on Software Testing and Analysis 2015 Jul 13 (pp. 234-245). ACM.

[22] A. Marchetto, MM. Islam, W. Asghar, Susi A, Scanniello G, “A multi-objective technique to prioritize test cases,” IEEE Transactions on Software Engineering. 2015 Dec 22;42(10):918-40.

[23] D. Di Nucci, A. Panichella, A. Zaidman, De Lucia A, “Hypervolume-based search for test case prioritization,” In International Symposium on Search Based Software Engineering 2015 Sep 5 (pp. 157-172). Springer, Cham.

[24] S. Wang, S. Ali, Yue T, Bakkeli Ø, Liaaen M, “Enhancing test case prioritization in an industrial setting with resource awareness and multi-objective search,” In Proceedings of the 38th International Conference on Software Engineering Companion 2016 May 14 (pp. 182-191). ACM.

[25] JA Parejo, AB. Sánchez, S. Segura, Ruiz-Cortés A, Lopez-Herrejon RE, Egyed A, “Multi-objective test case prioritization in highly configurable systems: A case study,” Journal of Systems and Software. 2016 Dec 1;122:287-310.

[26] Y. Bian, Z. Li, Zhao R, Gong D, “Epistasis based aco for regression test case prioritization,” IEEE Transactions on Emerging Topics in Computational Intelligence. 2017 Jun;1(3):213-23.

[27] S. Ali, Y. Li, Yue T, Zhang M, “Uncertainty-Wise and Time-Aware Test Case Prioritization with Multi-Objective Search,” Simula Research Laboartory, Tech. Rep. 2017.

[28] M. Khanna, N. Chauhan, Sharma D, Toofani A, Chaudhary A, “Search for prioritized test cases in multi-objective environment during web application testing,” Arabian Journal for Science and Engineering. 2018 Aug 1;43(8):4179-201.

[29] D. Pradhan, S. Wang, S. Ali, Yue T, Liaaen M, “Employing Rule Mining and Multi-Objective Search for Dynamic Test Case Prioritization,” Journal of Systems and Software. 2019 Mar 29.

[30] DB Mishra, R. Mishra, AA Acharya, Das KN, “Test Case Optimization and Prioritization Based on Multi-objective Genetic Algorithm,” In Harmony Search and Nature Inspired Optimization Algorithms 2019 (pp. 371-381). Springer, Singapore.