Multi-objective Test Case Minimization using Evolutionary Algorithms: A Review

Vandana
M.Tech Scholar, CSE Dept.
DCRUST Murthal, India
vndnghlwt16@gmail.com

Ajmer Singh
Asst. Prof, CSE Dept.
DCRUST Murthal, India
ajmer.saini@gmail.com

Abstract— **Software testing is one of the primal phase in various software development lifecycle models and consumes approximately 70% of development time and 40% cost of the overall budget. Nowadays automated testing tools along with different meta-heuristic algorithms which work similarly as simple testing techniques but they significantly outperforms when the complexity of the program is high are used in software testing phase to reduce the effort and time to test various program codes. Recent studies shows that various Evolutionary Algorithms (EA) like Artificial Immune System (AIS), Particle Swarm Optimization (PSO), Simulated annealing, Artificial Bee Colony (ABC), Cuckoo Search Algorithm (CSA), Ant colony optimization (ACO) are being functionalized in the field of Software Engineering to obtain optimal solutions. This review paper demonstrates the minimization of test cases using these evolutionary algorithms.**

Keywords- Software testing; Multi-objective optimization; Test case minimization ; evolutionary algorithms

1. INTRODUCTION

Developing software to test the software is referred as test automation or automated testing and in simple terms automated testing is automating the manual testing process. Test case prioritization is an important task when test cases are being executed while test cases must be prioritized as if in any condition the test ends precipitately so in that situation the best possible solution can be achieved. Prioritization of test cases also makes sure that the critical problems can be resolved in early stage and the most crucial test cases are executed before others. To achieve certain performance goals test cases are scheduled before hand in a sequence to improve their effectiveness and this is achieved using test case prioritization techniques. [2]

One of the major objectives in test case minimization is to reduce the redundant and obsolete test cases during testing so as to faster the application under test by correcting faults and removing errors in the beginning to lower the chances of failures. Recent studies shows that various Evolutionary Algorithms (EA) like Artificial Immune System (AIS), Particle Swarm Optimization (PSO), Simulated annealing, Artificial Bee Colony (ABC), Cuckoo Search Algorithm (CSA), Ant colony optimization (ACO) are being functionalized in the field of Software Engineering to obtain optimal solutions. Nowadays, test case minimization and test case generation on multi-objectives are trending in research community rather than working on single objectives.

Two types of major testing strategies are most likely to be used in software testing which are White box and Black box testing. White box testing aim is to determine a method that go through internal functionality to figure out the possible bugs and errors. [1] White box testing is further divided into two parts: static testing and dynamic testing. It is also known as structural testing. It has certain test data inputs which are used to test all the code paths. Black box testing or functional testing refers to a kind of testing in which we have set of inputs and expected outputs and this set of inputs is transformed into expected outputs by the software and this whole process is performed internally and unaware of how this process take place. Various techniques of performing black box testing are cause-effect graphing technique, boundary value analysis, decision table based testing and equivalence class testing. [3]

The properties defined under test adequacy criterion are statement coverage, functional requirements coverage, path coverage, code complexity and code coverage etc. There are three dimensions in testing criteria: type of code being considered, feedback use and source of information. [2][4][6]

According to researchers working in this field, it has been concluded that use of meta-heuristic algorithms along with optimization can lead to decrease in the count of redundant test cases and an increase in accuracy of automated testing as selection and recombination processes are faster in these meta-heuristic algorithms. [17]

2. MULTI-OBJECTIVE OPTIMIZATION TECHNIQUE

A multi-objective optimization problem in mathematical terms:

$$\mathbf{Min}\ (\mathbf{f_1(x)}, \mathbf{f_2(x)}, \dots, \mathbf{f_k(x)}) \tag{1}$$

$s.t. x \in X,$

$$f: X \to R^k, f(x) = (f_1(x), \dots, f_k(x) \tag{2}$$

Here function f is the vector-valued objective function while minimizing the negative of a function inverse the function can be maximized. $Y \epsilon R^k$ is used to denote the image of X. [7]

2.1 Pareto Dominance

A vector $u = (u_1, u_2, \ldots u_k)$ is said to dominate another vector $v = (v_1, v_2, \ldots v_k)$

2.2 Pareto Optimality

$$v = F(x') = (f_1(x'), f_2(x') \tag{3}$$

dominates

$$u = F(x) = (f_1(x), f_2(x), \ldots, f_k(x)) \tag{4}$$

2.3 Pareto Optimal Set

For F(x), the Pareto Optimal Set P* is formulated as :

$$P^* = \{x \in \Omega \mid \neg\exists x' \in \Omega\, F(x') \preccurlyeq F(x)\}. \tag{5}$$

2.4 Pareto Front

Pareto Front PF* is formulated as :

$$PF^* = \{F(x) \mid x \in P^*\}. \tag{6}$$

3. Components of Evolutionary Algorithms

Evolutionary algorithms are nature inspired algorithms and have different basic components that are used to obtain optimized solution for large number of particles, individuals, procedures or components

- Component 1: Representation of individuals
- Component 2 : Quality Measure or Fitness Function
- Component 3: Population
- Component 4: Parent selection mechanism
- Component 5: Recombination and Mutation
- Component 6: Replacement mechanism

3.1 Representation of individuals

The first step in defining an Evolutionary algorithm is representation of individuals and in this process mapping of phenotype onto genotype is performed while forming a bridge between the real world and the computational world. Here phenotype is the actual observable physical characteristics or traits of an individual while genotype consists of DNA genes which are responsible for particular characteristics or traits. [8]

3.2 Quality measure or Fitness Function

The fitness function or the quality measure function is used to represent the improvements in the selection mechanism. This functions objective is to convert problem into evolutionary context and is composed from a quality measure in the phenotype space and assign it to a genotype space. In evolutionary computing this function is also known as evaluation function. [8]

3.3 Population

Population is the basic unit of evolutionary mechanism where it represents the possible solutions and selection of optimal solution is carried out using evolutionary algorithms. Defining a population means specifying the population size or how many individuals are present in the spatial structure. In parent selection and survivor selection process the whole current population is selected and choices are made accordingly and the best individual is selected. The number of different possible solutions present specifies the diversity of a population. Other statistical measures taken in account during parent selection from the given solution are fitness value, entropy, no. of genotypes or no. of phenotypes. [8]

3.4 Parent Selection Mechanism

Parent selection mechanism is a probabilistic approach where individuals are distinguished on the basis of their quality measure function or fitness function and are selected as parents for the next generation while applying certain variations to create best possible offspring with improved quality. Individuals are selected randomly to avoid local optimum and certain selection methods are adopted which prefer best fit solutions and certain less fit solutions are also selected to maintain the diversity in the population. [8]

3.5 Recombination and Mutation

Based on how many individuals are chosen as input the variation operators are divided into two types: Mutation and Recombination operators. Mutation is basically alteration in the genes where it is operated on a genotype which is the input and a slightly changed offspring is delivered as output. Recombination operator or commonly known as crossover as the name suggests is used to combine two parent genotypes in a random manner to produce genotypes of the offspring. [8] [9]

3.6 Replacement Mechanism

Replacement mechanism or the survivor selection method is used to differentiate individuals on the basis of their fitness or quality function. It is similar to the selection process but with one difference that it is used when the offspring are being created from the selected parents from a large population size. The main factor affecting the choice is the fitness value where the parents with higher quality are favored and a deterministic solution is obtained.[8]

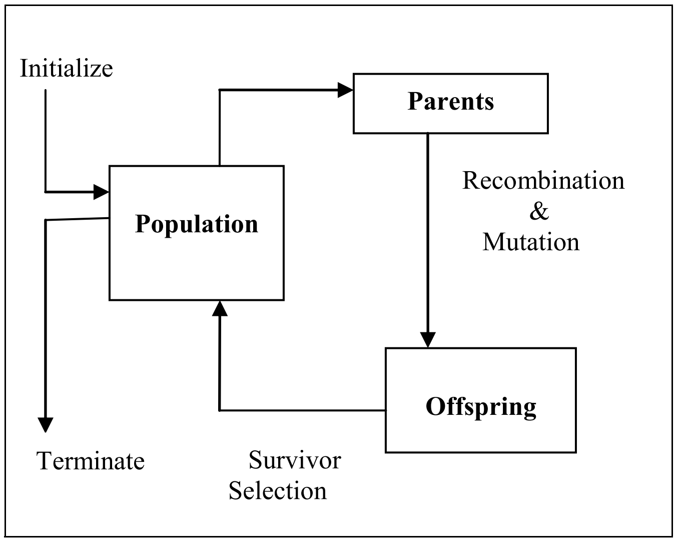

Figure 1.Components of Evolutionary Algorithms [8]

4. DIFFERENT EVOLUTIONARY ALGORITHMS

An evolutionary algorithm is a branch of computational intelligence where meta-heuristic optimization algorithms are used to obtain optimal solutions for a larger search space. EAs firstly select better individuals from large population size and then differentiate them based on the fitness measures, whole population is selected and choices are made where the best individuals are selected to create offspring. These individuals are then merged and altered to obtain new generation. [8]

4.1 Artificial Bee Colonisation

Artificial Bee Colonization model is an optimization technique in which a set of bees called swarm posses intelligent behavior. The initial searching of food is done by employed bees who are dedicated only to find the food source and then pass on the source and information to the onlooker bees by performing a waggle dance. The selection part of selecting good food source which is rich in high quality is carried out by onlooker bees that watch the dance and then decide the better source of food. When the source of food for the employed bees is vacant then these bees are called scout bees which again go in the search of food source. In this way the algorithm works where amount of nectar is equivalent to fitness associated and no. of employed bees is equivalent to no. of solutions in the population.

4.2 Particle Swarm Optimization

In computational intelligence branch of computer science PSO is a technique referred to provide an optimized solution for a random number of possible inputs or problems. Here the swarm is a chaotic collection of individuals that seems to move in random directions but that tend to form a cluster or group together. In this algorithm problem is solved by applying mathematical formulae to the search space in which particles randomly move in cluster and velocity and position of the cluster is depicted using graphs and accordingly the best and optimized solution is obtained. Here the particle movement is decided by local best position and is moved towards group best position and with change in time and search space they are modified for better position. Hence using this optimization algorithm PSO the best solution for the swarm is obtained. [4]

4.3 Artificial Immune system

Artificial Immune systems algorithm in artificial intelligence is adopted from biological immune system of vertebrate and intelligently models the characteristics in computer science engineering by means of computer simulators. The main characteristics of this immune system are the learning and memory skills which help in problem solving mechanism in this computational scenario. The basic techniques or type of algorithms under this AIS are: Negative Selection, Clonal Selection, Dendritic Cell and Immune Network algorithms.

4.4 Simulated Annealing

Simulated annealing technique is mostly used in computational task where the motive is to minimize or maximize certain parameters from random solutions. The term annealing is inherited from a metallurgical process in engineering where controlled heating and cooling of a substance take place under the supervision of engineers to lower the defects in the substance being annealed. Like other EAs this algorithm also provides optimal best solution on the basis of requirements given as input. [10]

4.5 Ant Colony Optimization

Ant Colony optimization is a technique inspired by natural world phenomena where randomly wandering ants go in search of food and when they find the source of high quality food they come back leaving pheromone trails behind for other ants of their colony. If other ants find the already traced and searched path then they follow the pheromone trails and collect food instead of finding to some other place but with time pheromone trail starts evaporating so the main aim of the ant scout is to find the higher density pheromone and that too traversing minimum distance possible. The pheromone density becomes higher on shorter paths than longer ones because the shorter paths are traced more frequently than the longer ones.[12]

4.6 Cuckoo Search Algorithm

Developed in 2009 CS is an algorithm discovered by Deb and Yang. Cuckoo search algorithm is inspired by a bird called Cuckoo and application area of this optimization algorithm are software engineering, pattern matching, data generation, in operating systems for job scheduling, prioritization of data, wireless sensor networks, artificial engineering. This algorithm is adopted by the researchers because of the basic characteristics of this bird cuckoo in egg laying and breeding system. Recently researchers have adopted this technique along with combinatorial testing for detection of faults and test case minimization. [13]

5. Analysis of Related Work

5.1 Review of various techniques:

The table shown below shows the brief review of techniques used by researchers for multi-objective test suite minimization

Table 1: Analysis of research papers

S.NO.	TITLE	TECHNIQUE USED	REF. NO.
1.	Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing	Cuckoo search for combinatorial testing	13
2.	Search Based Test Suite Minimization for Fault Detection and Localization: A Co-driven Method	NSGA-II	5
3.	Scope-aided test prioritization, selection and minimization for software reuse	Scope-aided test prioritization, selection and minimization technique	17
4.	Applying Ant Colony Optimization in software testing to generate prioritized optimal path and test data	Ant Colony Optimization	3
5.	Finding Representative Test Suit for Test Case Reduction in Regression Testing	Proposed genetic algorithm for test case reduction	18
6.	Minimizing Test Cases to Reduce the Cost of Regression testing	Genetic algorithm for test case reduction	19
7.	Optimization of test cases by prioritization using Genetic Algorithm	Test case Optimization using Genetic Algorithm	20
8.	Interaction-Based Test-Suite Minimization	Combinatorial Test Design	21
9.	Multi-objective test suite minimization using quantum-inspired multi-objective differential evolution algorithm	Quantum-inspired Multi-objective differential Evolution Algorithm	7
10.	On the Role of Diversity Measures for Multi-objective Test Case Selection	Asymmetric distance preserving	22
11.	A Multi-objective Particle Swarm Optimization for test case selection based on functional requirements coverage and execution effort	Particle swarm optimization (BMOPSO and BMOPSO-CDR)	4
12.	Application of Artificial Bee Colony algorithm to software testing	Artificial Bee Colony (ABC) based search algorithm	23
13.	Multi-objective Optimization based Differential Evolution Constrained Optimization Algorithm	Multi-objective Optimization based Differential Evolution for Constrained Optimization Algorithm	16
14.	MINTS: A General Framework and Tool for Supporting Test-suite Minimization	TEST SUITE MINIMISATION TECHNIQUES ALONGWITH PROPOSED MINTS TOOL	24
15.	A Non-Pheromone based Intelligent Swarm Optimization Technique in Software Test Suite Optimization	Artificial bee colony optimization (ABC) for test suite optimization	11
16.	Test Case Minimization and Prioritization Using CMIMX Technique	CMIMX Technique	2
17.	Generation of Pair wise Test Sets using a Simulated Bee Colony Algorithm	Pair wise testing a Combinatorial technique	25
18.	A Bi-objective Model Inspired Greedy Algorithm for Test Suite Minimization	Bi-Objective Greedy Algorithm (BOG)	26
19.	Design of Some Artificial Immune	Artificial Immune System	27

	Operators in Software Test Cases Generation	Algorithm	
20.	An Automated Test Case Generation Approach by Genetic Simulated Annealing Algorithm	Genetic Simulated Annealing Algorithm	28

5.2 Tools and software used in Test case minimization and optimization

Many tools have been developed in the literature, such as AETG, mAETG, PICT, CTE-XL, TVG, Jenny, TConfig, ITCH, IPO, IPOG, and IPOG-D.[14]

5.2.1 EVOSUITE

Many researchers use the automatically unit tests generating tool commonly for java software and the name of the tool is EvoSuite. It is also used for generating test cases to apply them on different testing modules. Here evolutionary algorithm is used for generating JUnit tests furthermore we have many plugins also that can be integrated in Maven, IntelliJ and Eclipse. This tool has been used on various open-source software for test case generation according to the user's requirement and finding thousands of vulnerabilities and potential bugs for several industrial systems. [14]

5.2.2 DEAP

Distributed Evolutionary Algorithms in Python (DEAP) is developed since 2009 at Université Laval and provide a framework for computation in evolutionary manner and also for testing different ideas. To implement most common evolutionary computation techniques such as PSO, genetic programming, genetic algorithm, evolution strategies, differential evolution it combines the tools required for the purpose and data structures. [15]

5.3 Literature Review

This review paper provides both detailed and survey analysis of trends in software testing where minimization of test cases and MOO is conceded using different EAs and the basis of this survey are previously published research papers in this field. Nowadays, test case minimization and prioritisation topics are of increasing importance in the research community as earlier only single objectives were solved to minimize the test cases and to prioritize them, but now focus is shifted to construct an automated testing method which resolves multiple objective simultaneously with same amount of cost and resources with the use of evolutionary algorithms. Charles Robert Darwin introduced the basic idea of natural selection on which these algorithms work, the evolutionary mechanism consist of selecting parents from a large population on the basis of fitness function parameter and performing various evolutionary methodologies thus evaluating new artificial evolutionary methodologies.

[8] A. E. Eiben. J.E.Smith. Introduction to Evolutionary Computing. Springer-Verlag Berlin Heidelberg, 2003

6. CONCLUSION AND FUTURE SCOPE

As stated relevant theory of multi-objective optimization and different evolutionary algorithms are used for test case minimization that too for solving multiple objectives, it has been concluded that use of meta-heuristic algorithms along with optimization can lead to decrease in the count of redundant test cases and an increase in accuracy of automated testing as selection and recombination processes are faster in these meta-heuristic algorithms.[16] We also observed that mostly test case minimization using evolutionary algorithms has approved keeping single objectives in mind (code coverage, branch coverage, fault detection, fault localisation, cost, effort, scheduled time etc) but the same can be applied for multiple objectives in future.

There are several possible research directions in the filed of test case minimization using Evolutionary Algorithms. It is observed that very little work has been carried out for test case minimization based on the parameters specific to object oriented testing like depth of inheritance, number of children etc. In future we would like to extend our work for test case minimization for object oriented testing

REFERENCES

[1] Rajiv Chopra. Software Testing: A Practical Approach 4th edition. S.K. Kataria &Sons, 2014

[2] Srivastava, P. R., Ray, M., Dermoudy, J., Kang, B. H., & Kim, T. H. (2009, December). Test case minimization and prioritization using CMIMX technique. In *International Conference on Advanced Software Engineering and Its Applications* (pp. 25-33). Springer Berlin Heidelberg.

[3] Biswas, S., Kaiser, M. S., & Mamun, S. A. (2015, May). Applying ant colony optimization in software testing to generate prioritized optimal path and test data. In *Electrical Engineering and Information Communication Technology (ICEEICT), 2015 International Conference on* (pp. 1-6). IEEE.

[4] de Souza, L. S., de Miranda, P. B., Prudencio, R. B., & Barros, F. D. A. (2011, November). A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort. In *Tools with Artificial Intelligence (ICTAI), 2011 23rd IEEE International Conference on* (pp. 245-252). IEEE.

[5] Geng, J., Li, Z., Zhao, R., & Guo, J. (2016, October). Search Based Test Suite Minimization for Fault Detection and Localization: A Co-driven Method. In *International Symposium on Search Based Software Engineering* (pp. 34-48). Springer International Publishing.

[6] Rothermel, G., Untch, R. H., Chu, C., & Harrold, M. J. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on software engineering*, *27*(10), 929-948.

[7] Kumari, A. C., Srinivas, K., & Gupta, M. P. (2012, December). Multi-objective test suite minimisation using quantum-inspired multi-objective differential evolution algorithm. In *Computational Intelligence & Computing Research (ICCIC), 2012 IEEE International Conference on* (pp. 1-7). IEEE.

[9] 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.

[10] Ramzi, Z. The Construction of Sequences for Identification of Digital Circuits using Simulated Annealing.

[11] Mala, D. J., Kamalapriya, M., Shobana, R., & Mohan, V. (2009, July). A non-pheromone based intelligent swarm optimization technique in software test suite optimization. In *Intelligent Agent & Multi-Agent Systems, 2009. IAMA 2009. International Conference on* (pp. 1-5). IEEE.

[12] Nagar, R., Kumar, A., Singh, G. P., & Kumar, S. (2015, February). Test case selection and prioritization using cuckoos search algorithm. In *Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE), 2015 International Conference on* (pp. 283-288). IEEE.

[13] Ahmed, B. S. (2016). Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing. *Engineering Science and Technology, an International Journal*, *19*(2), 737-753.

[14] Fraser, G., & Arcuri, A. (2011, September). Evosuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering* (pp. 416-419). ACM.

[15] Fortin, F. A., Rainville, F. M. D., Gardner, M. A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*(Jul), 2171-2175.

[16] Zhao, M., Liu, R., Li, W., & Liu, H. (2010, December). Multi-objective optimization based differential evolution constrained optimization algorithm. In *Intelligent Systems (GCIS), 2010 Second WRI Global Congress on* (Vol. 1, pp. 320-326). IEEE.

[17] Miranda, B., & Bertolino, A. (2016). Scope-aided test prioritization, selection and minimization for software reuse. *Journal of Systems and Software*.

[18] Mohapatra, S. K., & Pradhan, M. (2015, September). Finding representative test suit for test case reduction in
regression testing. In *Computer, Communication and Control (IC4), 2015 International Conference on* (pp. 1-6). IEEE.

[19] Mohapatra, S. K., & Prasad, S. (2014, March). Minimizing test cases to reduce the cost of regression testing. In *Computing for Sustainable Global Development (INDIACom), 2014 International Conference on* (pp. 505-509). IEEE.

[20] Jacob, T. P., & Ravi, T. (2013). Optimization of test cases by prioritization.

[21] Blue, D., Segall, I., Tzoref-Brill, R., & Zlotnick, A. (2013, May). Interaction-based test-suite minimization. In *Proceedings of the 2013 International Conference on Software Engineering* (pp. 182-191). IEEE Press.

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

[23] Dahiya, S. S., Chhabra, J. K., & Kumar, S. (2010, April). Application of artificial bee colony algorithm to software testing. In *Software Engineering Conference (ASWEC), 2010 21st Australian* (pp. 149-154). IEEE.

[24] Hsu, H. Y., & Orso, A. (2009, May). MINTS: A general framework and tool for supporting test-suite minimization. In *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on* (pp. 419-429). IEEE.

[25] McCaffrey, J. D. (2009, August). Generation of pairwise test sets using a simulated bee colony algorithm. In *Information Reuse & Integration, 2009. IRI'09. IEEE International Conference on* (pp. 115-119). IEEE.

[26] Parsa, S., & Khalilian, A. (2009, December). A bi-objective model inspired greedy algorithm for test suite minimization. In *International Conference on Future Generation Information Technology* (pp. 208-215). Springer Berlin Heidelberg.

[27] Ye, J., Zhan, Z., Zhang, Z., Dong, W., & Qi, Z. (2008, November). Design of some artificial immune operators in software test cases generation. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for* (pp. 2302-2307). IEEE.

[28] Li, B. L., Li, Z. S., Zhang, J. Y., & Sun, J. R. (2007, August). An automated test case generation approach by genetic simulated annealing algorithm. In *Natural Computation, 2007. ICNC 2007. Third International Conference on* (Vol. 4, pp. 106-111). IEEE.