Journal of Software Engineering and Applications, 2011, 4, 550-557
doi:10.4236/jsea.2011.49064 Published Online September 2011 (http://www.SciRP.org/journal/jsea)

Towards Multi-Faceted Test Cases Optimization

Manoj Kumar[1], Arun Sharma[2], Rajesh Kumar[3]

[1]Department of Computer Application, Galgotias University, Greater Noida, India; [2]Department of Computer Sc., Krishna Institute of Engineering and Technology (KIET), Ghaziabad, India; [3]School of Mathematics & Computer Application, Thapar University, Patiala, India.
Email: m_pachariya1@yahoo.com, arunshama2303@gmail.com, rakumar@thapar.edu

Received July 20th, 2011; revised August 23rd, 2011; accepted August 31st, 2011.

ABSTRACT

The target of software engineering is to produce high quality software product at low cost. Software testing is labour-intensive, ambiguous and error prone activity of software development. How to provide cost-effective strategies for software test cases optimization problem such as classification, minimization, selection, and prioritization has been one of the research focuses in software testing for a long time. Many researchers and academicians have addressed the effectiveness/fitness and optimization of test cases, and obtained many interesting results. However, one issue of paramount importance in software testing i.e. the intrinsic imprecise and uncertainty of test cases fitness, fitness parameters, multi-objective optimization, is left unaddressed. Test cases fitness depends on several parameters. Vagueness of fitness of test cases and their fitness parameters have created the uncertainty in test cases optimization. Cost and adequacy values are incorporated into multi-faceted optimization of test cases. This paper argues test cases optimization requires multi-faceted optimization in order to adequately cater realistic software testing. In this paper, authors have identified several parameters for test cases fitness and multiple objectives for test cases optimization. In addition above, authors have formulated the test cases optimization problem in three different ways using multi-faceted concept. These formulations can be used in future by authors and researchers.

Keywords: *Multi-Faceted Optimization*, *Test Cases*, *Test Data Adequacy Criteria*, *Test Case Fitness*

1. Introduction

Software testing is the process of exercising the programs with specific intend of finding errors prior to deliver. Although software testing is a very human-intensive, time consuming and itself an expensive activity, yet launching of software without proper testing may lead to cost potentially much higher than that of testing, specially in systems where human safety is involved [1,2]. Test cases are the inputs to the program under test. A test case is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Test cases pool may contain some redundant, irrelevant and unfit test cases. Since, testing is very expensive activity, unnecessary execution of redundant, irrelevant and unfit test cases will increase unnecessary burden of cost. The solution is to choose the fittest test cases and removing the unfit, redundant unnecessary ones, which in turn leads to test cases optimization [3-5]. Measuring fitness of test cases is always a daunting task. The term “fitness” refers to the appropriateness of test cases to check the quality of software. A test suite is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours. Multi-criteria test cases fitness evaluation, multi-objective test cases optimization may be the crucial problem for next generation software testing sorority. It requires to device next generation technologies to solve multi-faceted test cases optimization problem. The effectiveness of this verification and validation process depends upon the number of errors found and rectified before releasing the system. This, in turn, depends upon the fitness and number of test cases exercised. So, test cases optimization is necessary.

An optimization problem is the problem of finding the best solution from all feasible solutions. Multi-objective optimization (MO) also known as multi-criteria or multi-faceted or multi-attribute optimization is the process of simultaneously optimizing two or more conflicting objectives subject to certain constraints. The objective of MO optimization is to find the set of acceptable solutions and present them to the decision maker to take decision. Test cases optimization is the problem of finding the best subset (class) of test cases from a pool of test cases to be audited. It will meet all the objectives of testing concurrently. Most of the researchers evaluated the fitness of test cases only on single parameter fault detecting capability. Though, the fitness of test case depends on several parameters but consideration of only one parameter is not appropriate. Though, there are several objectives of test case optimization such as execution time, cost of data, cost of risks, and coveragebility, discussed in details in Section 3. But some objectives of test cases optimization are conflicting in nature, coveragebility of one objective will suffer other objective like cost, fitness of test cases and number of test cases in suite while considering all objectives concurrently. However most of test cases optimization approaches found in the literature are single objective and focused on the problem of maximizing the attainment of adequacy value without taking cost into account. It was not until 2001 that detailed empirical study was presented taking cost into account [6,7]. It was single objective optimization of test cases incorporating coveragebility and cost in ratio. Two objective test case optimization can be solved in such manner. If we want to optimize the test cases for three or more objectives, this approach is not fruitful. So it is not appropriate to estimate fitness of test cases just on single parameter and classify, select, prioritize the test cases on single objective. Single objective formulation of test cases fitness and optimization are not meeting the objectives of testing. Test cases should be optimized in such a way that it will achieve maximum of code coverage, maximum requirements coverage, high fault detecting capability, maximum mutant killing score and so far. The objective of test cases optimization is to reduce the number of test cases in suite to be audited and improve the effectiveness/fitness of test cases. So, test cases fitness evaluation, classification and selection of test cases should be treated as multi-faceted concept. It will surely reduce the cost & efforts of software testing and improve the quality of testing and reduce the number of test cases to be audited also.

2. Single Objective Formulation for Test Cases Optimization

This section briefly reviews test cases minimization, selection, classification, and prioritization, which collectively form part of the more general topic of test cases optimization. This section formalizes relationship between test cases classification, selection, minimization, prioritization and filtration that form the test cases optimization. Test cases minimization is a selection of smallest subset the test cases from a pool of test cases to be audited for a program. It covers as many program elements as the entire pool does. Test suite reduction seeks to reduce the number of test cases in a test suite while retaining a high percentage of the original suite's fault detection effectiveness. Test suite minimization techniques seek to reduce the effort required for regression testing by selecting an appropriate subset of test suite. Test suite minimization is minimal set cover problem, which uses greedy approximation approach to solve it. So, Test suite minimization is NP-complete problem [8,9]. Peculiar nature problems are those problems which require curious mix of data and knowledge driven approach to solve it. In search based software engineering, test cases optimization is a search space problem, which requires hybridization of data driven and knowledge driven approach to find near optimal solution of the problem. Hence, Test cases optimization is also peculiar nature problem [6,10]. Test cases selection is also finding minimal cardinality subset of test cases from the pool of test cases. One major difference between test cases minimization and test case selection is that test case selection chooses a temporary subset of test cases, whereas test suite minimization reduces the test suite permanently based on some external criterion such as structural coverage. Test case prioritization techniques try to find an ordering/ranking of test cases, so that some test case adequacy can be maximized as early as possible. Test case prioritization and filtration depend on quality of initial population of test cases. Selection and prioritization of test cases are the two important solutions to the problem of test case optimization. Test case filtration and prioritization are closely related. In fact, test cases can be filtered by selecting the first N ordered test cases. Therefore, any test case prioritization algorithm can be used as a test case selection algorithm. Naturally, it is desirable to select those test cases that are most likely to reveal defects in the program under test. When testing a program, software testers have to define the testing objectives first. A test suite is then constructed to satisfy the all objectives of testing. It is generally agreed that a test suite must achieve maximum coveragebility of all objectives of testing [11,12]. Usually, the constructed test suite may contain redundant test cases. A test case in a test suite is said to be redundant if the same testing objective can still be satisfied by other test cases of the test suite. Since the execution of test cases and evaluation their results are very expensive, it is of paramount importance to remove redundant test cases within a test suite. However, removal of all redundant test cases is practically infeasible because the problem is NP-complete. However, a weakness of test suite reduction is that the removal of some test cases from the test suite may potentially reduce the fault detecting capability of the test suite too [13,14]. To be worthwhile, the sum of the cost of test cases filtration, execution and audit of selected test cases should be less than the cost of that of all of the test cases of the original pool. The goal of test cases filtration is to chunk/filter out irrelevant, redundant and less fit test cases from the test suite. Test cases filtration is to chunk out subset of closely related test cases, so that a large portion of the defects would be found as if the whole test suite was to be used. It is often desirable to filter a pool of test cases for a program in order to identify a subset that will actually be executed and audited at a particular time. When it is uncertain that how many test cases can be run and audited, it is advantageous to order or rank the test cases as per priorities, so that the tester will select the test cases as per their rank or order, which permit tester to start quick, early fixing the most of the defects [11]. Single objective formulation of test cases minimization, selection and prioritization are as follows:

Definition 1 (Test Case Minimization): Let *T* be a set of test cases and *R* be a set of test objectives. *M* is a test case minimization of *T* with respect to R if and only if *M* is a subset of *T* that maximizes *R*(*T*).

Traditionally, *R* is defined with respect to a set $\{r_1, r_2, \cdots, r_n\}$ of syntactic elements of the program under test to be covered by the test suite, such that *R*(*X*) is defined to be the number of elements in $\{r_1, r_2, \cdots, r_n\}$ covered by *X*. Typically, we seek a set *M* that is smaller than *T*, one that is, therefore, a proper subset of *T*. Ideally, *M* will be minimal; no other minimized test suite will be smaller. In some approaches, we may seek to cover all elements of $\{r_1, r_2, \cdots, r_n\}$, rather than merely maximizing their coverage. When R is defined in terms of coverage, test case minimization becomes an instance of the set cover optimization problem. Test cases selection problem can be formulated in terms of test cases minimization problem. It becomes minimal set cover optimization problem. So, test cases minimization is NP-Complete problem.

Definition 2 (Test Case Selection): Let P_0 be a modified version of *P* and let *T* be a test suite. Let *CA* be a function which takes a pair of programs and reports the set of program elements of the form *A* that are different in P_0 compared to *P*. In this context, *A* plays the role of test adequacy criterion, substituted by values such as <branch adequacy> and <statement adequacy>. Let $RA(X, P, P_0)$ be the number of elements of $CA(P, P_0)$ covered when P_0 is executed on *X*. A test set T_0 is an A-adequate test case selection of *T* with respect to (P, P_0) if and only if T_0 is a test case minimization with respect to $RA(T_0, P, P_0)$.

Definition 3 (Test Case Prioritization): Let *T* be a test suite containing n elements and $T = \langle t_1', t_2', t_3'', \cdots, t_n' \rangle$ be a sequence on *T*. Let *F* be a function from test suite elements to some domain on which the relation ≥ imposes a total order. *T*' is a test case prioritization of *T* with respect to *F* if and only if for all *i*, $1 \le i \le n-1$, $F(t_1') \ge F(t_{i+1}')$. That is, *F* is monotonic over *T*.

3. Sketch of Multi-Faceted Test Case Optimization

Tester will desire to find the subset of test cases that accomplish multi-objectives concurrently in order to maximize the value obtained from inherently expansive process of executing several test cases, investigate the output produced by them. The test problem specification includes three main parts, the purpose of testing, test coverage criteria and the test strategy that will be employed. Testing objectives can be categorized into two classes which fall on adequacy value or fitness value (those to be maximum) and those which fall on cost side (those to be minimum). Minimization objective can usually be inverted to convert it into maximize objective. Some objectives have natural fitness function, also known as constraint that delimits the set of valid test cases. In search based engineering, these constraints are treated as objectives. In most realistic testing scenarios there will be at least one cost side and one value-based objective. There are several constraints and may be used as test objectives [15]. Review of existing literatures have brought out several parameters for assessing the fitness and objectives of test cases optimization like maximum number of defect detecting capability, minimum test design efforts/cost, minimum testing cost/execution cost, maximum coveragebility of client requirements/codes, minimum test execution time/effort, and maximum mutant killing score, etc are objectives and contributing as parameters for assessing the fitness of test cases. These parameters values/scores may be maximum/minimum according to the test objectives category. Details of test cases fitness evaluation parameters are given in **Table 2**. Contribution of test case parameters towards their fitness is vague and imprecise. Importance of testing objectives is also fuzzy. So, Concurrent consideration of all fuzzy parameters and testing objectives creates uncertainty. All fitness parameters/characteristics of test cases are not important for each project and they are not contributing equally to fitness of test cases.

3.1. Objective and Parameters Identification

Existing literatures are the evidence that there are several objectives for test cases optimization and fitness evaluation, whose details are as following (**Table 1**).

3.1.1. Cost-Oriented Objectives

The quality/fitness of the test cases is not only concern but cost is also one of the essential criteria, the whole purpose of test case optimization is to achieve more efficient testing in terms of the cost. Selection time and exe-

Table 1. Multi-objective test cases optimization.

S.No.	Objective Category	Objectives
1	Fitness Oriented Objectives	Code coverage Data Flow Based Control Flow Based Fault detection with fault Severity Fault detection Capability Mutation Killing Capability Client Requirement Execution Time Execution Efforts Fault Localization
2	Cost Oriented Objectives	Cost Benefit Analysis Execution Cost Design Cost Data Access Cost Setup Cost Third Party Cost
3	Project Change Volatility Objectives	Rate of Change in Requirements Increase in Testing Efforts
4	Optimization Constraint Objectives	Superiority Concurrency Exclusivity Dependability

Table 2. Summary of multi-objective test cases optimization parameters.

	Parameters	Sub-Parameters
Software Test Cases Fitness Evaluation Parameters	Fault Detecting Capability	Error Seeding Mutant Killing ability Fault Severity
	Control Flow Based Adequacy	Statement Coverage Branch Coverage Path Coverage Loop Coverage Relational Operator Coverage Table/Array Coverage
	Data Flow Based Adequacy	All Definition Criteria All Uses Criteria
	Efforts	Total Efforts Design Efforts Selection Efforts Execution Efforts Requirement Change Impact Analysis on Execution Efforts Efforts Benefits
	Cost	Total Cost Data Access Cost Setup Cost Design Cost Selection Cost Execution Cost Requirement Change Impact Analysis on Execution Cost Cost Benefits/Cost Saving
	Requirement Coverage Capability	Critical Requirement Coverage Rare Requirement Coverage Least Requirement Coverage Rate of Change in Requirements

cution time are the important factors for test cases optimization. Software testing cost includes data access cost, third party software cost, technical resources cost, setup and simulation cost [1,2]. Execution time is a natural candidate for test cases optimization. Execution time is one realistic measure of effort. Physical execution time of test cases is hard to measure accurately. Measurement is confounded by many external factors; different hardware, application software and operating system. Execution time is clearly a pressing concern for a tester, given the short build cycles within which they will typically have to perform the testing activities [7,16]. Data access cost concerns to databases software. Access to databases determines the coverage of the application under test. The population of the database will significantly affect the effectiveness of testing. We shall prefer realistic test cases rather than a mocked up version of the database, into which data may be systematically, but nonetheless synthetically added. Such synthetic data can lead to many false positives, because integrity constraints are not handled by the automated synthetic test generation algorithm. It can also lead to many false negatives. Data access cost includes cost of real data population (Human cost or payment to data provider), data retrieving cost. Some systems interact with third parties software, creating significant testing costs. There may be a price for accessing the systems of a third party. However, without such third party service access, it may not be possible to test the system fully [17]. Embedded systems are tightly coupled with their environment. These systems may consume resources that have a non-trivial cost. Cost of technical resources used in testing, is important cost driver. There may be setup costs associated with certain test cases. Setup cost includes cost of required devices, services, files. However, the setup costs for the test case may be significant in time and other costs. Such setup costs may also introduce dependencies, leading to an interaction between objectives and constraints. Testing automotive software is very expensive and requires simulation. The efforts of developing or deploying a simulation of the real system constitute a significant cost. Cost oriented objectives should be minimized in test case optimization problem [15].

3.1.2. Adequacy Value-Oriented Objectives

Software test adequacy criteria are the rules to determine whether a software system has been adequately tested, which points out the central problem of software testing *i.e.* "what is a test data adequacy criterion?" Number of test data adequacy criteria has been proposed and investigated in the literature like code coverage or control flow-based test adequacy criteria, data flow based adequacy criteria, fault-based adequacy criteria, and error-based criteria. The control-flow based adequacy criteria includes statement coverage, branch coverage, path coverage, Length-i path coverage, loop coverage, relational operator coverage, table coverage. Data-flow based adequacy criteria includes all definitions criterion, all uses criterion [18,19]. Fault Sensitive model is used to measure the fault detecting capability with fault severity. Test cases that reveal more likely categories of faults are more likely to be selected or prioritized. Fault History Sensitive model is used to measure the fault detecting capability using fault history. There is no guarantee that past fault revelation confers an on-going value to test cases. Such previously most fault-revealing test cases may be regarded as "proven star performers". Fault-based adequacy criteria include error seeding and mutant coverage or mutant killing score. So, fault detecting capability of test cases is very important and should be incorporated in test cases optimization [11,12,14]. Software testing is human-interactive activity. Tester, project manager, quality officer and customers are important player for software testing, each having their own experience, knowledge, psychology and opinions on testing priorities. Human psychological, sociological, knowledge have high impact on software testing. Business sensitive is highly subjective factor for testing. All features of software are not equally important. There are also more quantifiable business objectives. Business objectives are key concern for test cases optimization problem. Business Value Measurement (BVM) is a measure in which high importance or maximum weight values are assigned to business/customer critical requirement. Importance of business/client requirements are the vital objectives for test cases optimization problem. Test cases belonging to critical business/client requirements should be executed first [2,20].

3.1.3. Project Change Volatility (PCV)

PCV is based on the how many times consumer is modifying the project requirements during the software development cycle. PCV is one of the criteria which help to assess the requirement changes after the start of the implementation. High PCV increases the test efforts significantly and make it difficult to complete the project on time. PCV is also important for test case optimization. Test cases of high PCV value are first executed [12].

3.1.4. Multi-Objective Optimization Constraints

Most optimization problems involve various factors influencing the optimal results. In multi-objective test cases optimization problem, the factors that have an effect on the optimal solution are superiority, concurrency, exclusivity, dependability of test cases. In superiority constraint, some test cases have to be performed before others because they establish a system state in which the subsequent test cases become possible, or for which these later tests perform better in some way. We may treat this as an objective. In conjunction constraint, tests cases may be conjoined such that executing one, entails executing another. Such constraints may be soft (it is advantageous to the tester to test these two together) or hard (these two must be executed together). In exclusivity constraint, two test cases may be mutually exclusive. For instance, if one test completely exhausts a resource that is required by another, then these two tests cannot both be performed. Once again, these constraints may be soft or hard, but where they are present, the Test cases optimization process must take them into account. Otherwise, the test case selection and prioritization results produced by test cases optimization may not be viable. In dependence constraint, Inclusion of one test case may affect the cost of another. For instance, if we undertake the work required by a complex setup process for a certain test case, the same setup may be re-usable by other test cases. In this way there may be dependence between the cost of one test case and the costs of others. If we include one of the test cases, then the cost of all those that remain will be reduced; they share the same setup procedure and the costs associated with them [21,22].

3.2. Problem Formulations

Multi-Objective Optimization is defined as problem of finding the vector of decision variables X which satisfies the constraints and optimizes a vector function whose elements represent the objective functions. Generally it can be described as a vector function that maps a tuple of parameters (decision variables) to a tuple of objectives. The decision vector is also called the parameter space and the objective vector is also called the objective space. Find the vector $x^* = \left[x_1^*, x_2^*, \cdots, x_n^*\right]T$ which will satisfy the m inequality constraints: $g_i\left(x\right) \geq 0$ where $i = 1, 2, \cdots, m$ and the p equality constraints $h_i\left(x\right) = 0$ where $i = 1, 2, \cdots, p$.

Test cases optimization is multi-objective, NP-Complete, peculiar nature problem. Multi-objective test cases optimization is scalable and multi-modal optimization problem. It can be scaled to any number of objectives and constraints. Similarly, the multi-objective formulation of test optimization can be done in various ways. It can be formulated as multiple single objective functions, weighted sum approach, bottom-up approach and many others.

3.2.1. Multiple Single Objective Functions

First approach is most intuitive one and simple. In this approach, N different single-objective functions are used to construct a multi-objective test cases optimization problem. Different objective functions are simply used as different translation of single objective function. Details of objectives of test cases optimization are given in **Tables 1-2**.

$$\text{Maximum/Minimum} \quad Z_i = C_{ij} X_j$$

$$\text{Maximum} \quad Z_1 = C_{1j} X_j$$

$$\text{Minimum} \quad Z_2 = C_{2j} X_j$$

where $j = 1, 2, 3, \cdots, m$, and $i = 1, 2, \cdots, n$. N is the number of objectives and M is the number of constraints. Z_1 is fault detecting capability and Z_2 is execution cost of test case. It lacks Global optimal solution. Since the test case t_i optimal for objective function Z_1 is not optimal for objective function Z_2 and vice versa. It provides local optimal solution. It requires Pareto optimal set of solutions. Pareto optimal set contains individual optimal solution of each objective and trade off solutions of all objectives. It requires global Pareto optimal solutions.

3.2.2. Weighted Sum Approach

Second Weighted Sum Approach is common and simple. A set of n objectives, $O = \{O_1, O_2, O_3, \cdots, O_n\}$, and a set of weights $w_1, w_2, w_3, \cdots, w_n$ can be combined into a single weighted objective (WO),

$$WO(x) = w_1 * O_1(x) + w_2 * O_2(x) + \cdots + w_n * O_n(x)$$

where $w_1, w_2, w_3, \cdots, w_n$ are the weight values test objectives. Weight Values of test objectives can be estimated by using Fuzzy Logic based approach and will be explored in future. Details of objectives can be found in **Tables 1-2**.

3.2.3. Bottom-up Approach

In bottom-up approach, test cases optimization is considered as search space problem. Pareto optimal front is considered in this approach. Mathematical function describing Pareto optimal front is assumed in objective space. Pareto optimality is a notion from economics with broad range of applications in game theory and engineering. Pareto optimality provides a set of test cases not a single test case to be exercised on software under test. Pareto-optimal solutions are optimal in some sense. Therefore, like single-objective optimization problems, there exist possibilities of having both local and global Pareto-optimal solutions. Before we define both these types of solutions, we first discuss dominated and non-dominated solutions.

For a problem having more than one objective function (say, Z_i, $i = 1, 2, \cdots, N$ and $N > 1$). Any two solutions $x^{(1)}$ and $x^{(2)}$ can have one of two possibilities-one dominates the other or none dominates other. Solution $x^{(1)}$ is said to be dominate other solution $x^{(2)}$, if both the following condition are true.

1) The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives.

2) The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective.

If any one of above two is violated, solution $x^{(1)}$ does not dominate the solution $x^{(2)}$. It is also true that if $x^{(1)}$ dominates the solution $x^{(2)}$ then we can say $x^{(2)}$ dominated by the solution $x^{(1)}$ or $x^{(1)}$ is non-dominated by $x^{(2)}$. Between two solutions $x^{(1)}$ is non-dominated solution.

Based on this, the multi-objective optimization problem can be defined as the problem of finding a vector of decision variables x, which optimize a vector of N objective functions $f_i(x)$ where $i = 1, 2, \cdots, N$. The objective functions are the mathematical description of the optimization criteria. Without the loss of generality, it is assumed that the goal is to maximize f_i where $i = 1, 2, \cdots, N$. A decision vector x is said to dominate a decision vector y if and only if their objective vectors $f_i(x)$ and $f_i(y)$ satisfies:

$$\forall i \in \{1, 2, \cdots, N\}, \quad f_i(x) \geq f_i(y) \quad \text{and}$$

$$\exists i \in \{1, \cdots, N\}, \quad f_i(x) > f_i(y)$$

All decision vectors that are not dominated by any other decision vector are said to form the Pareto optimal set, while the corresponding objective vectors are said to form the Pareto frontier.

Above concept can be extended to find a non-dominated set of test cases form the pool of test cases. Considering the set of M test cases, each having N ($N >= 1$) objective function values. In Local Pareto Optimal set, if every member t_i in test suite Ts, there exist no test case t_j which dominates any member in the test suite Ts, then solution belonging to test suite Ts constitute local Pareto optimal test case. In Global Pareto Optimal set, if there exist no test case t_i in test suites space which dominates any member in the test suites space TS, then solution belonging to test suites space TS constitute Global Pareto optimal test case. The size and shape of Pareto Optimal fronts usually depends on number of objective functions and interaction among individual objective functions. If the objective functions are conflicting in nature to each other, resulting Pareto front may span larger than if the objective are more cooperating. However in multi-objective test cases optimization problem, some objectives are conflicting in nature to others. Resulting Pareto optimal front of multi-objective test cases optimization may contains many solutions, which can be found by using multi-objective optimization algorithm. When we consider the case of finding a set of non-dominated solutions rather than a single-point solution, multi-objective Evolutionary Algorithms (MOEA) have to perform a multi-modal search for global the Pareto-optimal set.

Now the multi-objective test cases optimization problem can be defined as follows:

Definition 4 (Multi-Faceted Test Cases Optimization)

Given: a vector of decision variables, x, and a set of objective functions, $f_i(x)$ where $i = 1, 2, \cdots, N$.

Problem: Maximize $\{f_1(x), \cdots, f_N(x)\}$ by finding the Pareto optimal set over the feasible set of solutions.

The multi-objective test suite minimization problem is to select a Pareto efficient subset of the test suite, based on multiple test criteria. It can be defined as follows:

Definition 5 (Multi-Objective Test Suite Minimization)

Given: a test suite, *T*, a vector of *M* objective functions, f_i, $i = 1, 2, \cdots, N$.

Problem: to find a subset of *T*, T_0, such that T_0 is a Pareto optimal set with respect to the set of objective functions, f_i, $i = 1, 2, \cdots, N$. The objective functions are the mathematical descriptions of test criteria concerned. A subset t1 is said to dominate t_2 when $(\{f_1(t_1), \cdots, f_N(t_1)\})$, the decision vector for t_1 dominates that of t_2.

The multi-objective test case selection problem is to select a Pareto efficient subset of the test suite, based on multiple test criteria. It can be defined as follows:

Definition 6 (Multi-Objective Test Case Selection)

Given: a test suite, T, a vector of N objective functions, f_i, $i = 1, 2, \cdots, N$.

Problem: to find a subset of *T*, T', such that T' is a Pareto optimal set with respect to the objective functions, f_i, $i = 1, 2, \cdots, N$. The objective functions are the mathematical descriptions of test criteria concerned. A subset t1 is said to dominate t_2 when the decision vector for t_1 $(\{f_1(t_1), \cdots, f_N(t_1)\})$ dominates that of t_2.

The multi-objective test case prioritization problem is to rank a Pareto efficient subset of the test suite, based on multiple test criteria. It can be defined as follows:

Definition 7 (Multi-Objective Test Case Prioritization)

Given: a test suite, *T*, a vector of *N* objective functions, f_i, $i = 1, 2, \cdots, N$.

Problem: to find a ranks of elements of *T*, such that T' is a Pareto optimal set with respect to the objective functions, f_i, $i = 1, 2, \cdots, N$. Let T' be a test suite containing *n* elements and $T' = \langle t_1', t_2', t_3'', \cdots, t_n' \rangle$ be a sequence on *T*. The objective functions are the mathematical descriptions of test criteria concerned. A subset t_1 is said to dominate t_2 when the decision vector for t_1 $(\{f_1(t_1), \cdots, f_N(t_1)\})$ dominates that of t_2.

4. Conclusion and Future Scope

Testing is complex, time consuming, human-intensive, full of uncertainty and expensive activity. Choosing the right fit test cases is an important and critical task in software testing. Literature is evident that there is no direct measure of fault revelation likelihood. There are different parameters for judging the fitness of test cases. Cost is also an important factor for test cases optimization involved in different ways and cannot be ignored. This complex interplay between cost and fitness/adequacy value is further compounded by the many additional validity constraints. These constraints, cost and fitness value are making test case minimization, selection and prioritization problems as multi-faceted optimization problem. The present paper advocacies that such a Multi-faceted approach for test cases fitness evaluation and test cases optimization is long overdue. In this paper, authors formulated the multi-faceted test cases optimization problem. It will be beneficial for researchers. In future, evolutionary and soft computing techniques will be explored for multi-faceted measurement framework for test cases optimization and fitness evaluation.

REFERENCES

[1] D. S. Alberts, "The Economics of Software," *Proceedings of National Computer Conference on Quality Assurance*, Montvale, Vol. 45, 1976, pp. 433-442.

[2] S. Hema and L. Williams, "On the Economics of Requirements Based Test Case Prioritization," *ACM SIGSOFT Software Engineering Notes*, Vol. 30, No. 4, 2005, pp. 1-3.

[3] Y. Shin and M. Harman, "Pareto Efficient Multi-Objective Test Case Selection," *Proceedings of the International Symposium on Software Testing and Analysis* (*ISSTA*), London, ACM press, 2007, pp. 140-150.

[4] D. J. Mala and V. Mohan, "ABC Tester—Artificial Bee Colony Based Software Test Suite Optimization Approach," *International Journal of Software Engineering*, Vol. 2, No. 2, 2009, pp. 15-43.

[5] D. J. Mala, V. Mohan, "Quality Improvement and Optimization of Test Cases—A Hybrid Genetic Algorithm Based Approach," *ACM SIGSOFT Software Engineering Notes*, Vol. 35, No. 3, 2010, pp. 1-14.

[6] R. Mohanty, V. Ravi and M. R. Patra, "The Application of Intelligent and Soft-Computing Techniques to Software Engineering Problems: A Review," *International Journal of Information and Decision Sciences*, Vol. 2, No. 3, 2010, pp. 233-272.

[7] S. Elbaum, A. G. Malishevsky and G. Rothermel, "Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization," *Proceedings of the 23rd International Conference on Software Engineering* (*ICSE*-01), IEEE Computer Society, May 2001, pp. 329-338.

[8] W. E. Wong, J. R. Horgan, A. P. Mathur and A. Pasquini,

"A Test Set Size Minimization and Fault Detection Effectiveness: A Case Study in Space Application," *Proceedings of IEEE* 21*st Annual International Computer Software and Application Conference* (*COMPSAC*-97), IEEE Press, Los Alamitos, 1997, pp. 552-528.

[9] S. Tallam and N. Gupta, "A Concept Analysis Inspired Greedy Algorithm for Test Suite Minimization," *The Sixth ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering*, New York, ACM Press, 2005, pp. 35-42.

[10] P. McMin, "Search-Based Software Test Data Generation: A Survey," *Proceedings in Software Testing*, *Verification and Reliability*, Vol. 14, 2004, pp. 105-156. doi:10.1002/stvr.294

[11] J. M. Kim and A. Porter, "A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments," *Proceedings of the* 24*th International Conference on Software Engineering*, Orlando, 2002, pp. 119-129.

[12] K.Varun, Sujata and M. Kumar, "Test Case Prioritization Using Fault Severity," *International Journal of Computer Science and Technology* (*IJCST*), Vol. 1, No. 1, September 2010, pp. 67-71.

[13] J. A. Jones and M. J. Harrold, "Test-Suite Reduction and Prioritization for Modified Condition/Decision Coverage," *IEEE Transactions on Software Engineering*, Vol. 29, No. 3, March 2003, pp. 195-209. doi:10.1109/TSE.2003.1183927

[14] W. E. Wong, J. R. Horgan, S. London and A. P. Mathur, "Effect of Test Set Size Minimization and Fault Detection Effectiveness," 17*th International Conference on Software Engineering* (*ICSE*'95), Seattle, 23-30 April 1995.

[15] H. Mark and J. Wegener, "Getting Results with Search-Based Software Engineering: Tutorial," 26*th IEEE International Conference and Software Engineering* (*ICSE* 2004), Los Alamitos, 2004, IEEE Computer Society Press, pp. 728-729.

[16] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer and R. S. Roos, "Time Aware Test Suite Prioritization," *International Symposium on Software Testing and Analysis* (*ISSTA* 06), ACM Press, Portland, 2006, pp. 1-12.

[17] M. Bozkurt, M. Harman and Y. Hassoun, "Testing Web Services: A Survey," Technical Report TR-10-01, Department of Computer Science, King's College London, April 2010.

[18] Z. Hong, "Axiomatic Assessment of Control Flow-Based Software Test Adequacy Criteria," *Software Engineering Journal*, Vol. 10, No. 5, September 1995, pp. 194-204.

[19] D. Leon and A. Podgurski, "A Comparison of Coverage-Based and Distribution-Based Techniques for Filtering and Prioritizing Test Cases," *Proceedings of the* 14*th International IEEE Symposium on Software Reliability Engineering* (*ISSRE*'03), Denver, 2003, pp. 442-453.

[20] Y. Shin, M. Harman, P. Tonella and A. Susi, "Clustering Test Cases to Achieve Effective and Scalable Prioritisation Incorporating Expert Knowledge," *ACM International Conference on Software Testing and Analysis* (*ISSTA* 09), Chicago, 19-23 July 2009, pp. 201-212.

[21] E. Emelie, P. Runeson and M. Skoglund, "A Systematic Review on Regression Test Selection Techniques," *Information & Software Technology*, Vol. 52, No. 1, 2010, pp. 14-30. doi:10.1016/j.infsof.2009.07.001

[22] L. Zheng, M. Harman and R. Hierons, "Meta-Heuristic Search Algorithms for Regression Test Case Prioritization," *IEEE Transactions on Software Engineering*, Vol. 33, No. 4, 2007, pp. 225-237. doi:10.1109/TSE.2007.38