Test Case Reduction Techniques - Survey

Marwah Alian
Princess Sumaya University for Technology
Amman, Jordan

Dima Suleiman
Princess Sumaya University for Technology
Amman, Jordan

Adnan Shaout
The University of Michigan - Dearborn
Michigan - Dearborn

***Abstract*—Regression testing is considered to be the most expensive phase in software testing. Therefore, regression testing reduction eliminates the redundant test cases in the regression testing suite and saves cost of this phase. In order to validate the correctness of the new version software project that resulted from maintenance phase, Regression testing reruns the regression testing suite to ensure that the new version. Several techniques are used to deal with the problem of regression testing reduction. This research is going to classify these techniques regression testing reduction problem.**

Keywords—Regression testing; Test case reduction; Test Suite

I. INTRODUCTION

Regression testing is done to ensure the validity of modified software which is an essential activity related to making maintenance. The goal of this activity is to ensure that bug fixes and new functionality do not harm the correct functionality inherited from original program [1]. The size of the test-suite grows when new test cases are added to the test suite which increases the cost of regression testing [2]. Moreover, regression testing requires running a large program on a large number of test cases which means it can be expensive in both human and machine time [1].

Reducing cost is the target of researchers on the use of test-suite reduction techniques. Therefore, a number of different methods have been studied to deal with test suits such as minimization, selection and prioritization. Test suite minimization or reduction aims to reduce the number of tests to run. [3]

However, the main objective of most of proposed algorithms is to reduce the test suite size. In [4] they use the concept of Set theory to minimize the larger test suite. The set is defined as a collection of objects and entities. One set for each object type is created, and set operations such as Cartesian product, union, intersection and difference are used to reduce the size of test suite. This paper depends on the power of set functions.

While in [5] they combine the concept of software testing and Case-Based Reasoning (CBR) by assuming that test cases are treated as cases in CBR and they discuss how to maintain a number of test cases in software testing using the Case-Based Maintenance (CBM) if there is a set of test cases generated by the Path-Oriented technique. They propose a number of maintenance techniques that are used for removing unnecessary test cases and for controlling the growth of test cases. The process of maintaining CBR is called Case Based Maintenance (CBM) which is a policy of adding, updating, and deleting cases. Nicha et al. concentrate on the Deletion Policy for CBM. Their experiments show that the proposed technique can be used to reduce the number of test cases required for software testing in an efficient way.

Regression test suite (RTS) can be reduced using model-based regression which based on an Extended Finite State Machine (EFSM), for each elementary modification (EM) data and control dependences are used to capture interaction between EFSM transitions and three interaction patterns. Reduced RTS only includes test cases that one of their 3 interactive patterns is not produced for any other test case [6]. Requirements changes lead to modifications in the EFSM, Modification in model result in performing three types of model based regression testing: testing the model affection in modification, testing the modification affection on the model and testing the effect of modifications on unmodified part of the model. An extension of EFSM is the system under test (SUT) that support variable ranges over several data types and operations. In each transition level a changes occurs in EFSM. Changes can add or delete transitions [7].

The selection of most appropriate regression method is not easy since everyone has its advantages and disadvantages. Adaptive regression testing (ART) can be used to determine the most effective one which take into consideration the organization situation and testing environment [8]. ART uses certain criteria to determine which regression testing to use such as cost and benefits in addition to the organization situation such an evaluation have a problem called multiple criteria decision making (MCDM) problem, one of MCDM methods is Analytical Hierarchy Process (AHP), the results of the experiment show that techniques selected by AHB is cost effective. Decision maker must define a hierarchy that contains a description of a problem to be solved in order to be able to use AHP. Hierarchy contains the goal and the factors that may be used by decision maker.

The purpose of this survey is to collect and consider papers that deal with one of regression testing techniques that are test suite reduction. Our intention is to provide a state-of-the-art view on this field. Many different approaches have been proposed in order to reduce the cost and time of regression testing.

The rest of this research is organized as follows: Section II gives a review about other surveys in the field. Section III defines the problem of test suite reduction. In section IV a detailed outlook on the classification of test suite reduction techniques and section V presents a summary for the test suite techniques and discussion.

II. Related Work

Detecting faults in the program is the objective of software testing and it provides more assurance on the quality of the software. As the software evolves the size of the test suite grows because of the addition of new test cases to the test suite [9]. It is important to develop techniques to minimize available test suites because of the time and resource constraints for re-executing large test suites. [10] Hundreds of techniques have been proposed to solve the problem of test suite reduction, and several surveys have been introduced for such techniques.

In software testing field a survey of Chaurasia et al. [11] is given which is a literature review of test case reduction techniques. They discuss and compare three algorithms presented in [12][13][14]. All papers are about test case reduction but Control Flow Graph is used in all the methods. These algorithms worked on the basic Basis Path Testing, which is the first method worked on the low level of the code, exceptions, conditions, and loops. They discuss the algorithm advantages, disadvantages, cost, and time. In the three algorithms, time, cost and the number of test cases are reduced significantly.

In [15] they present a survey of using Genetic Algorithm in different software testing techniques. They describe how GA works and the applications of GA in different types of software testing like test planning, minimization of test cases in regression testing, model based testing and web testing. They discuss one technique for each type of software testing except for model based testing they discuss four techniques. In regression testing they introduce the technique proposed by [16] and their comparison with vector based technique in test suite reduction. Also, they compare GA Parameters that are used in different types of software testing. GA parameters are Crossover Rate, Mutation Rate, and Number of Generation.

While in [17] an empirical study of five different regression test optimization techniques is presented. They give a brief description for the three regression testing optimization techniques that are selection, prioritization, and minimization. Then they describe five implemented algorithms that are slicing algorithm, incremental algorithm, genetic algorithm, adaptive firewall approach, and simulated annealing. Also, they introduce a comparison that is based on different qualitative and quantitative criteria such as execution time of tests, precision, number of tests selected, user parameters, global variables handling, and type of testing.

A review is made to compare between Heuristics, Genetic Algorithms and Linear programming based techniques. Heuristics produced smaller size reduced set it work very good on it but it must be optimized in large scale test suite, on the other hand Heuristic GRE produced optimal representative set. Genetic Algorithm concerns about the block based test suites on software and coverage, in Genetic algorithm with time constraints the test suite becomes smaller results in minimizing running time. Smallest set is produced in Integer Linear Programming compared with other algorithms. For significant test suite reductions two or more techniques can be combined to form hybrid techniques [18].

In [19] various test case generation methods are presented such as minimizing, selection, prioritization and evaluation test cases. Also many methods that help test engineers in scheduling and ranking test cases are discussed such as test cases prioritizations and selection techniques. Some of test case generation depends on application such as those generated for web, object oriented, UML, evolutionary applications and structured based systems. System Requirements and use cases can be used to derive test cases; also test cases can be generated using UML sequence diagram, these test cases are generated for object oriented programs. In order to be able to find the sub optimal test cases we can use genetic algorithms to let test cases meet certain criteria such as if the test cases are adequate to statement, branch, and path coverage. Code based test generation generates test cases from a code, four types of inputs are taken: program to be tested, -and some information related to runtime. Dynamic path testing and evolutionary technique generate test cases by using many test values in program execution; test cases can be prioritized according to the shortest path. Test case also can be generated by traversing the graph from parent root to child node in Graph Traversal Algorithm, by using breadth or depth first in a graph tree.

There are different representations of graph models proposed for procedural programs which are discussed in [20] survey such as Control Flow Graph (CFG), Program Dependence Graph (PDG) and System Dependence Graph (SDG). Flow graph is a directed graph a set of nodes represents program statements, there are two nodes called start and stop nodes and there exists path from a start to every other node in flow diagram and from there to stop node. Data Dependence Graph is used to represents relationships between elements of a program. The relationship can be either data or control dependency.

There are different representations of graph models proposed for procedural programs that are discussed in [20] survey such as Control Flow Graph (CFG), System Dependence Graph (SDG) and Program Dependence Graph (PDG) . Flow graph is a directed graph in which a set of nodes represents program statements, there are two nodes called start and stop nodes and there exists path from a start to every other node in flow diagram and from there to stop node. Data Dependence Graph is used to represents relationships between elements of a program. The relationship can be either data or control dependency.

However, other surveys has a narrow view for the field and many types of techniques about test suit reduction requires classification into specific groups or classes represents similarities and common properties. In this survey we classify test case reduction techniques into: requirement based, coverage based, program slicing, genetic algorithm, greedy algorithm, hybrid algorithm, clustering and fuzzy logic.

III. Test Suite Reduction Problem

According to the definition given by [21], test suite reduction problem can be can be defined as:

Given a test suite T represents a set of test cases {t1, t2, t3,…, tn}, a set of test requirements R = {R1, R2, . . . , Rn} to be covered, and subsets of T, S= {S1, S2, . . . , Sn}, where each test set is associated with Ri. The objective is to find the representative subset RS ⊆ S that satisfies all of requirements and has at least one test case for each requirement R.

IV. Test Suite Reduction Techniques

Regression testing is defined as a software maintenance activity which is done to ensure the proper functionality of the software. Test suits that are developed during the development phase have a large size that it is not possible to run the entire test suite due to the time and cost. Therefore, regression test reduction process is advisable in order to reduce the test suite to minimal set of test cases that will cover all the faults in minimal time [22]. In this survey we consider the proposed test suite reduction techniques and their classification.

As shown in Fig. 1 test case reduction techniques are classified into: requirement based, coverage based, slicing, genetic algorithm, greedy algorithm, hybrid algorithm, clustering and fuzzy logic.

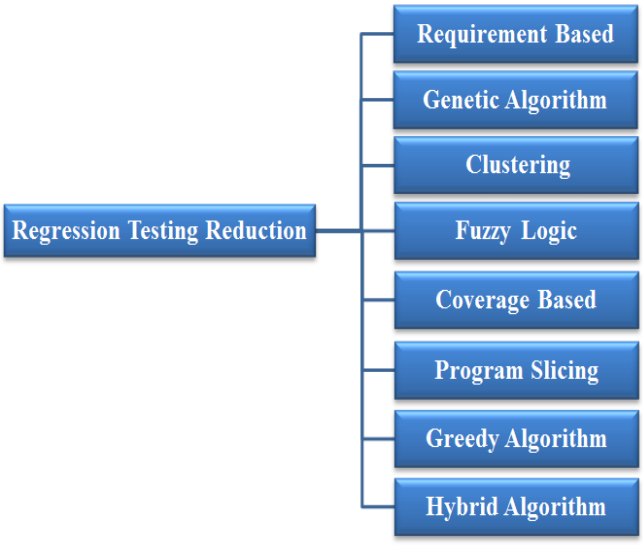

Fig. 1. Regression testing reduction techniques

A. Genetic Algorithms

There are many issues of software testing like effective generation of test cases, test case reduction, prioritization of test cases, etc. These issues demand on effort, time and cost of testing. The use of evolutionary algorithms for automatic test case generation and reduction has been an interest for researchers [15]. One such form of evolutionary algorithms is Genetic Algorithm (GA) that is computational Intelligence based approach used as a solution for the problem of test cases reduction like evolutionary computation.

In [2] they propose a genetic algorithm, for test-suite reduction which builds the initial population based on test history, it calculates the fitness value using coverage and cost information then using genetic operations it selects the successive generations. These steps will be repeated until a minimized test-suite set is found. The results show that the proposed test-suite reduction technique has cost-effectiveness and generality.

Also, the research of [23] investigates the use of genetic algorithms, for test-suite reduction. They propose a model that builds the initial population based on test history, but it calculates the fitness value depending on coverage and run time of test cases, the fit tests only allowed to be in the reduced suite. This generational process is repeated until an optimized test-suite is found.

In [16] they define the time-aware regression testing reduction problem and propose a genetic algorithm for this problem. This work describes parent selection, crossover and mutation processes of the genetic algorithm. The redundancy of test cases is removed from regression testing suite by the proposed algorithm they also minimize the total running time of the remaining test cases.

While in [24] a new approach for test case reduction is presented and implemented. This approach depends on genetic algorithm technique with varying chromosome length in order to reduce test cases in a test suite by finding a representative set of test cases that fulfill the testing criteria.

Approaches that based on Genetic algorithms need to further study the fault detection capability of block based test suite or coverage based test suite or other criteria [25].

B. Requirement Based

The purpose of test suite reduction is to satisfy all testing requirements with a minimum number of test cases. Before generating test cases it is necessary and possible to optimize testing requirements.

In [26] they solve test suite reduction by testing requirement optimization. They proposed a requirement relation graph which is proposed to minimize the requirement set by graph contraction. The experiment is designed and implemented using specification-based testing. In this work, testing requirement relation graph is introduced in order to hide the details of testing requirements and test cases and to achieve test suite reduction. Some requirement contraction methods are proposed to generate a small requirement set. Also, an empirical study on specification-based testing is performed. The study compares the relative effectiveness of testing requirement optimization to test case reduction. In order to compare with test suite reduction, all test cases of each testing requirement are generated then the greedy algorithm is applied on the constructed test suites for reduction.

The ability to detect faults is reduced as a result of reducing the size of test suite. The redundancy in test suites and size can be reduced using a model checker based techniques to create test cases [27]. The main idea of this approach is that not all test cases in test suite are important to achieve requirements, so that a subset of test cases can be chosen that fulfill the requirements. Model-checker take a finite state model and temporal logic property as input and as a result counter example will be returned if the property is not fulfilled. Counter example is a sequence of states from the initial state to the violating one. Since removing of test cases from test suite

have negative impact on the ability to detect faults, this can be avoided by transforming the test cases so the redundant parts can be omitted. Common prefix of the test cases is specified then the only part after the prefix of test cases is interesting. And if there is another test case that ends with the same prefix we talked about previously then we can append the remainder of the first test case to the second test case and omit the first one. In this research the quality of test suite after removing the redundant test cases does not suffer, the experiments show that the reduction is significant and one drawback is the run time complexity.

The basic classic needs of test suite reduction is to reduce the time and cost and maintain the effectiveness of fault localization, so that a technique that keep the system free from errors is needed and at the same time keeping the efficiency by reducing the number of test cases[28].

In [28] they study three techniques and the best out of them is determined, the first one is test suite reduction and requirement optimization, if there is a test cases in input domain that satisfy testing requirements then the testing requirements is feasible. The second one is dynamic domain reduction (DDR); these algorithms based on effective testing in detection of errors, there are many testing methods such as: path testing, the purpose of this algorithm is not to test every path. Another algorithm is independent program path where at least one new processing path is considered. DDR is based on constraint based testing. The third algorithm is Ping Pong technique which uses heuristic technique to reorder the test cases that provides a good solution but not the optimal one.

The requirement optimization is good when dealing with the finite Boolean expression that classifies the requirements as true of false test cases. When dealing with arrays, loops and expression DDR succeeds. Ping Pong assures the domain coverage but it is time consuming technique, expensive and more memory is needed. It takes requirements from natural language.

Some algorithms use the decision table concepts that depend on the requirement gathered from the user to create test cases [29]. A framework of this algorithm consists of three steps: functional requirements analysis and condition determination, input generation for rule development and testing, in this step decision table is generated and with corresponding rules and test cases ,and the last step is rule development and testing using opened rules, in this step the test output will be generated. The redundancy reduction of test case is up to 30% which save cost and time; the proposed algorithm does not require the tester to have knowledge of coding.

C. Fuzzy logic

Optimization of test suites can be achieved by using fuzzy logic. It is a safe technique and can reduce the regression testing size and execution time [30]. Fuzzy logic can be used in many areas such as communication, bio informatics and experts systems. Level of testing using fuzzy logic is based on objective function quite similar to human judgment.

A combination of fuzzy logic with genetic algorithm and swarm optimization can be used to make optimization in test suite for multi-objective selection criteria. Therefore, in [31] they aim to find a test suite that is optimal for multi-objective regression testing. They propose an expert system that use a technique and level of testing based on a defined objective function, similar to human judgment using fuzzy logic based classification.

While in 2013, [32][33] use some CI based approaches in order to optimize the test suite and analyze the test suite for safe reduction which can be estimated using control flow graphs. Test cases of optimal solutions are traversed on these graphs and it is found that only fuzzy logic is safe while other approaches will be inadequate for regression testing.

D. Clustering

When designing test cases, there are many redundant test cases with no use. Such redundancy increases the testing effort and increase the cost and time of testing.

In order to reduce the time spent in testing, the number of test cases is reduced. In [34] they use the data mining approach of clustering technique in software testing to reduce the test suite.

With the help of the clustering techniques the number of test cases is reduced and the efficiency of software testing is improved. By using clustering the program can be checked with any one of the clustered test cases rather than with the entire test case that is produced by the independent paths.

A new approach is discussed by [35] which divide the test cases into clusters according to the similarity in profiling. Previous researches made partitions on execution profile by representing it as a binary vector which contains only number of times the function executed without taking into considerations the sequence of execution, so that this paper provide enhancements by making three types of profiles: file execution sequence, function call sequence and function call tree. The results show that the relation between function calls and sequential information will make enhancements in detecting the faults.

Clustering techniques based on selecting test cases of using coverage and distribution based techniques. They produce smaller representative sets of test cases but less fault detection ability [25].

E. Coverage Based

An important issue to be considered during test suite reduction is the coverage aspect. Coverage-based reduction techniques have to ensure that majority of the execution paths of the given program are exercised [9].

Regression testing must preserve the fault detection in addition to minimizing the size and the time of test. In Case Base Reasoning (CBR) there are three cases classifications: case, auxiliary and pivotal case. Case based is Artificial intelligent which search for the most similar problems to solve problems; in this case a memory is needed. Auxiliary case can be deleted without affecting competence, but only reduces the efficiency of the system. On the other hand pivotal cases have a direct effect on the competence of a system if it is deleted. As we said CBR uses path coverage criteria to reduce the redundancy test cases. Path coverage uses a control flow graph

which can derived from a source code, path oriented test case generation technique used to derive a test cases from a control flow graph where each state can reveal a fault.

In [36] they use the case based reasoning (CBR) which is one of artificial intelligent concepts. This study introduces three methods. CBR and software testing have the same problems that there are many redundant test cases after reduction, the ability of reducing the faults decreases, and grow of test cases is uncontrollable. The key research issues of CBR are: continuous growing in CBR size is the existence of too many redundancy cases and the deletion all redundancy in CBR take a lot of time. In order to control number of cases there are many algorithms such as add, delete and maintenance approaches, deletion algorithms are the efficient algorithms to maintain the size of CBR system. [36] Treat the test cases as cases in CBR with the assumption that path oriented test cases method used to generate a set of test cases. There are two classifications of test cases reduction techniques proposed in this research, the first one is coverage based technique and the second one is concept analysis based technique. Their research propose three methods using CBR: The first method is Test Case Complexity for Filtering (TCCF), the proposed method that applies CBR concepts can be described in the following steps: Determine the coverage set, reachability set, auxiliary set from which we can compute the complexity for each test case and the last step is to remove test cases from the auxiliary set that have a minimum complexity value, the complexity of test case can be computed as follows: High when number of test cases are larger than the average number of test cases in test suites, when they are equals then the complexity is medium and when number of test case smaller than average it is considered as low.

The second algorithm is Test Case Impact for Filtering (TCIF) in this method the impact value is the impact of test cases according to ability to detect faults when test cases removed. The impact of test case is high when at least one fault of many times has been revealed by test cases, medium when the faults revealed for only one time and low when test case has never revealed the faults. The first three steps in this algorithm are the same as the ones in (TCCF) but the last step is different. The last step in TCIF is to remove test cases with minimum of impact value from the auxiliary.

The third algorithm is Path Coverage for Filtering (PCF) method; path testing is a structure testing since you choose test cases that determine the path to be taken within the program structure. This algorithm uses a coverage value which determines how many nodes that test case can cover. The procedure of this method has the following steps: identify the coverage set, calculate the coverage value which depends on number of test cases in each group and the last step is to remove all test cases with minimum coverage value.

The experiments randomly generate 2000 test data used in telecommunication industry. There are list of measures used in this experiment such as: number of test cases, reveals faults ability, and total reduction time. The results show that in PCF the number of test cases is minimized more than other algorithms and it consumes the least reduction time, but it is the worst in ability to reveal faults. While TCCF and TCIF are the best in term of faults detections, they are the worst that they require a lot of time in reduction process.

The main problem related to using these algorithms is that for huge systems the path coverage is ineffective since it needs a time and consumes a cost in identifying the coverage from a source code.

A technique called coverall algorithm is proposed in [37]. The test cases are generated automatically and the tester has no options to do that. This method uses algebraic conditions to give a value to variable, this variable value resulting in fewer numbers of test cases. Also this method can be used in loops and arrays, the reduction rate of test cases will reach 99%. In addition to creating test cases automatically, this method also minimizes the number of test runs.

To generate test cases automatically there are four steps: finding all constraints from start to finish node, identifying the variables with minimum and maximum values, test path to find constant values and creating a table of all possible values from the above values. As a result of this algorithm the number of generated test cases is smaller than many other algorithms such as Ping-pong, it also keeps the test cases generation to a single run. This technique is the best technique among many other techniques such as GetSplit algorithm, Ping-pong technique and Dynamic Domain Reduction (DDR) in term of reduction of test cases and other factors discussed before, however if there are more than two variables in program code the method is not applicable. This algorithm can be improved if the description of initial domain input is no longer required [37].

It is possible to reduce test suites without affecting the coverage of the states. In [38] they propose an algorithm that covers all reachable states in closed loop controller. The approach focuses on path coverage since it generate test cases from accessing the source code, a path is a sequence of statements from the beginning of program execution to its end while the sub path is a sequence from the beginning of a specified function execution to its end. In this approach test cases are generated depends on intuition that is, as long as it covers all sub-paths it is not necessary to cover all paths. There are two steps of this approach: Identify the test cases that cover all sub paths in program based on implementation of code and remove any test case that covers an already covered sub paths. For experiments purposes this approach applied to five controller programs for real world medical protocols, these programs are: PennNeuroICU, PennCardiac, PennMICU, PennHyper-Glycemia, and PennIntraoperative Experiments show that test case reduced by tens of thousands with no reduction in fault finding capability.

Reference [39] proposed a technique called TestFilter used for reducing test cases, it uses statement coverage. TestFilter choose non-redundant test case according to their weight, this process reduce the test cases storage management and execution cost.

F. Program Slicing

Program slicing is introduced by [40]. This technique is used to check a program over a specific property and to build a slice set, which is a set of statements effect to determine a statement; in many cases it is the output statement of a

program, based on input values. Slicing techniques can help to show control-flow of a program for each test case and it is important to specify which statements are invoked with that test case. [55]

There are three types of slicing techniques: static slicing [40][41][42], dynamic slicing [43][44], and relevant slicing [45]

In [1] a survey is made for seven approaches that use program slicing for reducing the cost of regression testing. There are three groups of these approaches; the first group includes those using dynamic slicing, while the second group includes approaches that use program dependence graph, and the third one includes those using data flow definition of slicing.

To reduce the cost of regression testing, the study of [46] uses two algorithms: the first one generates a program called differences, this algorithm called like this since it captures the difference between certified and modified program, where certified is the previously tested program without changes and modified is the program with modification. The second algorithm uses existing test cases to test components new in modified, also it uses the test cases for which modified and certified program produced the same outputs. The idea is to avoid the cost of using new test cases and to avoid rerunning test cases that produce the same output. The second algorithm uses a context slice which is one of new type of inter procedural slice. Inter procedural slice contains a program that includes components that capture all execution statements. The slice can be defined as follows: if we have a program component *p* and variable *x* then slice includes all statements in a program that causes effect on variable *x*.

Using slicing techniques can decrease the number of required test cases and consequently the cost and time of testing will be decreased.

However, in [47] they propose a method that is intended to reduce the cost and time of testing and they investigate the effect of slicing techniques on the reduction rate of testing cost and time. This method focuses on parts of the program code that have significant impact on its output while those parts of program that have no effect on the program output are eliminated from testing process. Hence, as the size of the code is reduced, testing time and cost will be decreased. Their experiments show that a large number of program instructions, branches and paths can be covered by a small number of test cases in the sliced program.

G. Greedy Algorithm

One of well-known heuristics proposed for code-based reduction is Greedy algorithm. This algorithm can also be applied on test suites obtained from Model-based techniques. It selects the test case which satisfies the maximum number of unsatisfied requirements and an arbitrary choice is made if there is a tie situation. This process is applied repeatedly to all test cases in the test suite and produces a reduced test suite. It is stopped after all test requirements are satisfied [48][49].

In [10] inspired greedy algorithm for test suite reduction is proposed. This algorithm is based on formal concept analysis of the relation between testing requirements and test cases. This analysis is used for objects that have discrete properties. They consider test cases as objects and requirements as their attributes. Using concept analysis framework, maximum grouping of objects and attributes are identified and called contexts. Reduction rules are used for reducing objects and attributes. This greedy algorithm differs than classical greedy heuristic which uses object implications without considering attribute implication. In their algorithm, context table is constructed initially then the size of context table is reduced by applying the objects reduction rules, attribute reduction and owner reduction. The reduction of objects and attributes will reduce the size of the context table by removing redundant attributes and objects from further consideration. While the owner reduction not only removes redundant attributes and objects but also it selects a test case which will be added to the reduced suite and the requirements covered by these test cases are not considered in further iterations. In each iteration, interference among test cases is also removed using greedy heuristic. The size of the reduced test case suite generated by their algorithm has the same or smaller size than that generated by the traditional heuristic algorithms.

Also, Weighted greedy algorithm is proposed by [50] for test suite reduction also called Weighted Set Covering Technique. This work starts by determining test cases which can satisfy all the requirements. If the test case does not satisfy requirements then the algorithm repeatedly eliminate redundant test cases then update the test suite and the remaining requirements that are uncovered. The essential test cases that are selected are added to the reduced set. In order to handle the remaining uncovered requirements, prioritization and sorting take place for test cases. Then, a selection for test cases depends on decreasing order of priority is repeated until all the requirements are satisfied. The optimized test suite had a higher efficiency. Their experiment is made on the test suite of Student Achievement Retrieval Navigation Model. The algorithm reduces the size of the test suites and minimizes test cost.

Real world java programs are used to implement four test suite reduction techniques for JUnit test suites [51]. The study of [51] cares about benefits and the cost of test suite reduction. The results show that JUnit suite is reduced without affecting the fault detection capability. The first technique involved is the greedy technique where a test case that satisfies maximum number of unsatisfied requirements is selected. The second technique is Harrold et al.'s Heuristic [21]; the main idea of this technique is to select test cases according to their essential. The next technique is GRE Heuristic brings together characteristics of essential test cases and 1 to 1 redundant test cases into greedy strategy, the heuristic terminates when all requirements are satisfied. The last algorithms is ILP Technique, the first ILP is single objective while the second ILP is multiple objective test suit reduction. The purpose of first ILP model is to minimize then number of test suite selected [51].

Coverage Based Test Suite Reduction (CBTSR) is a new algorithm for Test Suite Reduction that is proposed by [9]. They identify an optimal representative test set for test cases that are related to the given testing objective. Then they Apply

data flow testing to generate test cases as well as requirements in order to examine the physical structure of the program and to locate sub-paths. After that they use the proposed CBTSR algorithm for test suite reduction. Also, they perform a set of empirical studies on ten subject programs.

The reduction process in CBTSR algorithm starts with the construction of test case requirement matrix which maps the testing requirements with test cases. An association between a test case and requirement is represented by one or zero otherwise. Each row in the matrix denotes the requirement coverage and each column denotes the test case which overlaps with the requirement. Then the generation of the reduced test suite is made through simple mathematical operations. The results show that CBTRS algorithm selects near optimal test cases which satisfy a maximum number of testing requirements. Thus, it reduces the size of the test suite by retaining test cases that offer maximum percentage of Requirement Coverage.

Test suite reduction approaches that based on Greedy algorithm provide significant reduction in test suite but need to be optimized in large scale test suites [25].

H. Hybrid Algorithm

Some algorithms try to reduce the number of test cases using hybrid techniques such as genetic algorithms and bee colony [52]. Bee colony consists of three groups of bees: employed, onlookers and scouts. Using bees as agents the algorithm can explore the minimum set of test cases. This paper suggests using ant colony with genetic algorithms.

Moreover, in [53] they formulate three hybrid combinations, Rank, Merge, and Choice, and describe their usefulness. They produce a uniform representation for hybrid criteria and suggest that the hybrid criteria of others can be described using Merge and Rank formulations, and that the hybrid criteria outperform the constituent individual criteria.

While in [54] they propose multi-objective test suite reduction. They introduce a hybrid multi-objective genetic algorithm.

Their algorithm combine the efficient approximation of the genetic algorithm with the greedy approach to produce high quality Pareto fronts in order to achieve multiple objectives. Objective functions are considered as a mathematical description of test criterion. A cost cognizant version of the greedy algorithm is implemented for two objective optimization that are computational cost and statement coverage.

Three optimization objectives are also considered for fault detection history such as code coverage, fault coverage and execution time. These objectives are combined using the classical weighted-sum approach by taking the weighted sum of fault coverage per unit time and code coverage per unit time. The testing decisions that are taken by their technique have been more efficient.

V. SUMMARY

Regression testing is made when changes are performed to existing software in order to provide confidence that the new changes that are introduced do not affect the behavior of the existing, unchanged part of the software. As software evolves, the test suite tends to grow which implies that it may be expensive to execute the entire test suite [3].

Hundreds of techniques have been proposed in order to reduce the number of test cases, but there are still many researches in this field. In this research, a review for what have been proposed by researchers to solve the problem of test case reduction is presented and a classification for test case reduction is introduced. The reviewed test case reduction techniques are classified into: requirement based, coverage based, slicing, genetic algorithm, greedy algorithm, hybrid algorithm, clustering and fuzzy logic. More details about these techniques and their classification are demonstrated in Table II.

However, Greedy algorithm based techniques provide significant reduction in the number of test cases, but they need to be optimized in large scale test suites. While genetic algorithm based techniques need to be examined on the fault detection capability and other criteria.

While hybrid techniques are introduced for significant reduction in test case suites, they provide high complexity. Clustering techniques select test cases based on coverage and distribution techniques, they produce smaller representative sets but less fault detection ability [25].

The main problem related to coverage based techniques is that for large systems the path coverage is ineffective since it consumes time and cost in identifying the coverage from a source code [36]. The advantages and disadvantages of the proposed classifications are given in Table I. Many techniques can be incorporated with existing hybrid techniques and with genetic algorithms more mutation strategies can be introduced.

TABLE I. CLASSIFICATTION OF TEST SUITE REDUCTION- ADVANTAGES AND DISADVANTAGES

Classification	Advantages	disadvantages
Program slicing	decrease the number of required test cases and consequently the cost and time of testing will be decreased.	need to be examined on the fault detection capability and larger generated data
Genetic algorithm	Reduce the number of test cases and also decreases total running time.	need to be examined on the fault detection capability and other criteria
Greedy algorithm	provide significant reduction in the number of test cases	involve random selection of test case in a tie situation.

		need to be optimized in large scale test suites
Fuzzy logic	a safe technique and reduce the regression testing size and execution time	Need more experiments and studies
Requirement base	Provide a good percentage of redundancy reduction of test cases.	Some of them are time consuming and need more memory depending on how to represent the requirements
Coverage Base	reduction rate of test cases is very high and it reduce time	for large systems the path coverage is ineffective since it consumes time and cost in identifying the coverage from a source code
Hybrid algorithm	provide significant reduction in the number of test cases and multi-objective optimization	high complexity
Clustering	produce smaller representative sets of test cases	less fault detection ability

TABLE II. TEST SUITE REDUCTION TECHNIQUES

Year	List of Authors	Classification	Paper Title	Technique	Enhancement
1995	David Binkley	Program Slicing	Semantics Guided Regression Test Cost Reduction,	Two algorithms are used: the first one generates a program called differences, this algorithm called like this since it captures the difference between certified and modified program, were certified is the previously tested program without changes and modified is the program with modification. The second algorithm uses a context slice which is one of new type of inter procedural slice.	decrease the number of required test cases
2005	Xue-ying Ma, Bin-kui Sheng, and Cheng-qing Ye,	Genetic Algorithm	Test-Suite Reduction Using Genetic Algorithm	Builds the initial population based on test history, it calculates the fitness value using coverage and cost information.	it has cost-effectiveness and generality
2005	Sriraman Tallam, Neelam Gupta	Greedy Algorithm	A Concept Analysis Inspired Greedy Algorithm for Test Suite minimization	Inspired greedy algorithm for test suite reduction. Test cases are considered as objects and requirements as their attributes. Using concept analysis framework, maximum grouping of objects and attributes are identified and called contexts. Reduction rules are used for reducing the size of context table by applying the objects reduction rules, attribute reduction and owner reduction.	The size of the reduced test case suite has the same or smaller size than that generated by the traditional heuristic algorithms
2006	Preeyavis Pringsulaka and Jirapun Daengdej,	Coverage Based	Coverall Algorithm for Test Case Reduction	Coverall algorithm uses algebraic conditions to give a value to variable, this variable value resulting in fewer numbers of test cases. This method can be used in loops and arrays.	The reduction rate of test cases reach 99%.
2006	Saif-ur-Rehman Khan, Aamer Nadeem,	Coverage Based	TestFilter: A Statement-Coverage Based Test Case Reduction Technique	TestFilter uses statement coverage by choosing non-redundant test cases according to their weight.	This process reduce the test cases storage management and execution cost.
2006	B. Suri, I. Mangal, and V. Srivastava,	Hybrid Algorithm	Regression Test Suite Reduction using an Hybrid Technique Based on BCO And Genetic Algorithm	Combine genetic algorithms and bee colony. Bee colony consists of three groups of bees: employed, onlookers and scouts. Using bees as agents the algorithm can explore the minimum set of test cases.	Explore the minimum set of test cases.

2007	Gordon Fraser and Franz Wotawa ,	Requirement Based	Redundancy Based Test-Suite Reduction	Subsets of test cases that fulfill the requirements are chosen. Model-checker take a finite state model and temporal logic property as input and as a result counter example will be returned if the property is no not fulfilled.	The reduction is significant but one drawback is the run time complexity
2008	Zhenyu chen, baowen xu, xiaofang zhang, changhai nie.	Requirement Based	A novel approach for test suite reduction based on Requirement relation contraction.	a requirement relation graph is proposed to minimize the requirement set by graph contraction. An empirical study on specification-based testing is performed.	The study compares the relative effectiveness of testing requirement optimization to test case reduction
2010	Shin Yoo , Mark Harman.	Hybrid Algorithm	Using hybrid algorithm for Pareto efficient multi-objective test suite minimization	introduce a hybrid multiobjective genetic algorithm. Their algorithm combine the efficient approximation of the greedy approach with the genetic algorithm to produce high quality Pareto fronts in order th achieve multiple objectives.	The testing decisions that are taken by their technique have been more efficient.
2010	Siripong Roongruangsuwan and Jirapun Daengdej,	Coverage Based	Test Case Reduction Methods by Using CBR,	use an artificial intelligent concept of case based reasoning (CBR). propose three methods using CBR: Test Case Complexity for Filtering (TCCF), Test Case Impact for Filtering (TCIF) and Path Coverage for Filtering (PCF) Method	In PCF the number of test cases is minimized more than other algorithms and it consumes the least reduction time
2010	S. Nachiyappan, A. Vimaladevi and C.B. SelvaLakshmi,	Genetic Algorithm	An Evolutionary Algorithm for Regression Test Suite Reduction	initial population is built based on test history, but it calculates the fitness value depending on coverage and run time of test cases,	Reduce test case suite size.
2011	Lingming Zhang, Darko Marinov, Lu Zhang, Sarfraz Khurshid,	Greedy ALgorithm	An Empirical Study of JUnit Test-Suite Reduction	The study cares about benefits and the cost of test suite reduction by testing four techniques on JUnit test suites; greedy technique, Harrold Heuristic, GRE Heuristic and ILP.	The results show that JUnit suite reduced with affection the fault detection capability.
2012	Shengwei Xu, Huaikou Miao, Honghao Gao,	Greedy Algorithm	Test Suite Reduction Using Weighted Set Covering Techniques.	Weighted greedy algorithm is used for test suite reduction also called Weighted Set Covering Technique. It starts by determining test cases which can satisfy all the requirements. If the test case does not satisfy requirements then the algorithm repeatedly eliminate redundant test cases then update the test suite and the remaining requirements that are uncovered. The experiment is made on a test suite of Student Achievement Retrieval Navigation Model.	The optimized test suite had a higher efficiency. The algorithm reduces the size of the test suites and minimizes test cost.
2012	Liang You, Yansheng Lu	Genetic Algorithm	A genetic algorithm for the time-aware regression testing reduction problem	removes redundant test cases in the regression testing suite minimizes running time of the remaining test cases	Reduce the size of test suite and running time.
2012	Haider, A.A.; Rafiq, S.; Nadeem	Fuzzy logic	Test suite optimization using fuzzy logic	an expert system that use a technique and level of testing based on a defined objective function, similar to human judgment using fuzzy logic based classification	Optimize test suite.
2013	Christian Murphy, Zoher Zoomkawalla, Koichiro Narita,	Coverage Based	Automatic Test Case Generation and Test Suite Reduction for Closed-Loop Controller Software	An algorithm that covers all reachable states in closed loop controller. It focuses on path coverage since it generates test cases from accessing the source code. Two steps of this approach: Identify the test cases that cover all sub paths in program based on implementation of code and remove any test case that covers already covered sub paths. For experiments purposes this approach applied to five controller programs for real world medical protocols.	The number of test cases is reduced by tens of thousands with no reduction in fault finding capability.
2013	A. Ali Haider, A.	Fuzzy logic	Computational Intelligence	Use CI based approach and analyses the	fuzzy logic is a

	Nadeem, S. Rafiq,		and Safe Reduction of Test Suite	test suite for safe reduction which can be estimated using control flow graphs. Test cases of optimal solutions are traversed on these graphs and it is found that only fuzzy logic is safe while other approaches will be inadequate for regression testing	safe approach and it is adequate for regression testing
2013	Sampath, S.; Bryce, R.; Memon, A.M.	Hybrid Technique	A Uniform Representation of Hybrid Criteria for Regression Testing	Three hybrid combinations are formulated, Rank, Merge, and Choice, and describe their usefulness. They produce a uniform representation for hybrid criteria and suggest that hybrid criteria of others can be described using Merge and Rank formulations, and that the hybrid criteria outperform the constituent individual criteria.	Use Rank, Merge, and Choice operations to perform hybrid criteria that outperform the constituent individual criteria.
2013	Asghar Mohammadian , Bahman Arasteh.	Program Slicing	Using Program Slicing Technique to Reduce the Cost of Software Testing.	A method that focuses on parts of the program code that have significant impact on its output while those parts of program that have no effect on the program output are eliminated from testing process. It shows that a large number of program instructions, branches and paths can be covered by a small number of test cases in the sliced program.	As code size reduced, testing time and cost are decreased.
2014	Gupta, A. ; MNNIT Allahabad, Allahabad, India ; Mishra, N. ; Kushwaha, D.S	Requirement Based	Rule Based Test Case Reduction Technique Using Decision Table	A framework consists of three steps: functional requirements, analysis and condition determination. the proposed algorithm does not require the tester to have knowledge of coding.	The redundancy reduction of test case is up to 30% which save cost and time;
2014	B.subashini , d.jeyamala..	Clustering	Reduction of test cases using clustering Technique	Use data mining approach of clustering technique to reduce the test suite. By using clustering the program can be checked with any one of the clustered test cases rather than with the entire test case that is produced by the independent paths.	the number of test cases is reduced and the efficiency of software testing is improved.
2015	Sudhir Kumar Mohapatra, Srinivas Prasad	Genetic Algorithm	Finding Representative Test Case for Test Case Reduction in Regression Testing	Reduce test cases in a test suite by finding a representative set of test cases that fulfill the testing criteria.	Reduce the number of test cases.
2015	R. Wang, B. Qu, Y. Lu,	Clustering	Empirical study of the effects of different profiles on regression test case reduction	divide the test cases into clusters according to the similarity in profiling. provide enhancements by making three types of profiles: file execution sequence, function call sequence and function call tree.	The relation between function calls and sequential information will improve detecting the faults.
2015	Preethi Harris and Nedunchezhian Raju.	Greedy Algorithm	A Greedy Approach for Coverage-Based Test Suite Reduction.	The reduction process starts with the construction of test case requirement matrix which maps the test cases with the testing requirements. An association between a test case and requirement is represented by one or zero otherwise. Then the generation of the reduced test suite is made through simple mathematical operations.	It selects near optimal test cases that satisfy the maximum percentage of Requirement Coverage. and it reduce the test suite size

REFERENCES

[1] D. Binkley, The application of program slicing to regression testing, Information and Software Technology, 40(11-12), pp. 583-594.

[2] X. Ma, B. Sheng, and C. Ye, Test-Suite reduction using genetic algorithm,Vol. 3756 of the series Lecture Notes in Computer Science, 2005, pp. 253-262, springer.

[3] S. Yoo, and M. Harman, Regression testing minimization, selection and prioritization: a survey, Software Testing, Verification and Reliability. 22(2012), pp.67–120, DOI: 10.1002/stvr.430

[4] I. Mangal, D. Bajaj and P. Gupta, Regression Test Suite Minimization using Set Theory, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, No. 5, 2014, pp. 502-506

[5] N. Kosindrdecha, S. Roongruangsuwan and J. Daengdej, Reducing test cases created by path oriented test case generation, American Institute of Aeronautics and Astronautics, Inc. AIAA Infotech@Aerospace 2007 Conference and Exhibit, California, USA.

[6] Y. Chen, R. Probert, H. Ural, Regression test suite reduction using extended dependence analysis, Proceedings of the 4th international workshop on Software quality assurance, in conjunction with the 6th ESEC/FSE, ACM Press, 2007, pp.62-69.

[7] B. Guo, M. Subramaniam and H. Guo, An approach to regression test selection of adaptive EFSM Tests, Fifth IEEE International Conference on Theoretical Aspects of Software Engineering.2011. pp. 217 – 220.

[8] Md. Arafeen and H. Do., Adaptive regression testing strategy: An empirical study, 22nd IEEE International Symposium on Software Reliability Engineering, 2011.

[9] P. Harris and N. Raju, A greedy approach for coverage-based Test Suite reduction, The International Arab Journal of Information Technology, Vol. 12, No.1, 2015 pp. 17-23.

[10] S. Tallam, N. Gupta, A concept analysis inspired greedy algorithm for test suite minimization, 2005 ACM 1595932399/05/0009

[11] V. Chaurasia, Y. Chauhan and T. K, A survey on test case reduction techniques, International Journal of Science and Research (IJSR), 2012

[12] J. Offutt, Z. Jin and J. Pan, "The dynamic domain reduction procedure for test data generation," Software Practice and Experience, Vol. 29, No. 2, 1999, pp. 167-193.

[13] Q. Wang, S. Jiang and Y. Zhang, "An approach to generate basis path for programs with exception-handling constructs", In IACSIT Press, 2012, International Conference on Computer Science and Information Technology (ICCSIT), Singapore.

[14] Dr. R.P. Mahapatra, M. Mohan and A. Kulothungan, "Effective tool for test case Execution time reduction," In IACSIT, International Symposium on Computing, Communication and Control (CSIT), Singapore, 2011.

[15] C. Sharma, S. Sabharwal, R. Sibal, A survey on software testing techniques using genetic algorithm, IJCSI International Journal of Computer Science Issues, 2013, Vol. 10, No. 1, No 1.

[16] L. You, Y. Lu, "A genetic algorithm for the time-aware regression testing reduction problem", International conference on natural computation, IEEE, 2012, pp. 596 – 599.

[17] Jyoti and K. Solanki, A comparative study of five regression testing techniques : A Survey , INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH, Vol. 3, No. 8, 2014, pp. 76-80.

[18] R. Singh and M. Santosh, Test case minimization techniques: A review, International Journal of Engineering Research & Technology (IJERT), Vol. 2, No. 12, 2013. Pp.1048- 1056

[19] I. Hooda and R. Chhillar, A review: study of test case generation techniques, International Journal of Computer Applications. Vol. 107, No.16, 2014, pp. 33-37

[20] S. Biswas and R. Mall, Regression test selection techniques: A survey, Informatica 35. 2011, pp. 289–321

[21] M. Harrold, R. Gupt and M. Soffa, "A methodology for controlling the size of a test suite," ACM Transactions in Software Engineering and Methodology, Vol. 2, No. 3, 1993, pp. 270-285.

[22] Isha Mangal Deepali Bajaj Priyanka Gupta, Regression Test Suite Minimization using Set Theory , International Journal of Advanced Research in Computer Science and Software Engineering 4(5), May - 2014, pp. 502-506

[23] S. Nachiyappan, A. Vimaladevi and C.B. SelvaLakshmi, &ldquo, An evolutionary algorithm for regression test suite reduction, Proc. Int',l Conf. Comm. and Computational Intelligence, 2010, pp. 503-508.

[24] S. Mohapatra, S. Prasad, "Finding representative test case for test case reduction in regression testing", IJISA, vol.7, no.11, , 2015, pp.60-65. DOI: 10.5815/ijisa.2015.11.08

[25] R. Singh and M. Santosh, Test case minimization techniques: A review, International Journal of Engineering Research & Technology (IJERT). Vol. 2, No. 12. 1048 -1056.

[26] Z. chen, b. xu, x. zhang, c. nie, A novel approach for test suite reduction based on Requirement relation contraction, ACM. 390-394.

[27] G. Fraser and F. Wotawa, Redundancy based test-suite reduction, In Proceedings of the 10th International Conference on Fundamental Approaches to Software Engineering, Springer. Vol. 4422, 2007, pp. 291-305.

[28] R. SALWAN , R. SEHGAL, Test cases reduction technique considering the time and cost as evaluation standards, International Journal of Computer Science and its Applications, 2013, pp.347-351.

[29] A. Gupta, N. Mishraa and D. Kushwaha, Rule based test case reduction technique using decision table, 2014, pp. 1398 – 1405.

[30] Z. Anwar and A. Ahsan, Multi-objective regression test suite optimization with Fuzzy logic, IEEE. INMIC 2013.

[31] Haider, A.A.; Rafiq, S.; Nadeem, A. "Test suite optimization using fuzzy -logic", Emerging Technologies (ICET), International Conference on, 2012, pp. 1 – 6

[32] Haider, A.A.; Nadeem, A.; Rafiq, S. "Computational intelligence and safe reduction of test suite", Emerging Technologies (ICET), IEEE 9th International Conference on, 2013, pp. 1 – 6

[33] Haider, A.A.; Nadeem, A.; Rafiq, S. "On the Fly Test Suite Optimization with FuzzyOptimizer", Frontiers of Information Technology (FIT), 11th International Conference on, 2013, pp.101 – 106

[34] B.subashini, d.jeyamala, Reduction of test cases using clustering Technique. International Journal of Innovative Research in Science, Engineering and Technology Vol 3, Special Issue 3, 2014, International Conference on Innovations in Engineering and Technology (ICIET'14). 1992-1995

[35] R. Wang, B. Qu, Y. Lu, Empirical study of the effects of different profiles on regression test case reduction, IET Softw., 2015, Vol. 9, No. 2, pp. 29–38

[36] S. Roongruangsuwan and J. Daengdej, Test case reduction methods by using CBR, Assumption University, ceur-ws.org,Vol-646,2010.

[37] P. Pringsulaka and J. Daengdej.2006., Coverall algorithm for test case reduction. In Aerospace Conference, 2006 IEEE.IEEE.

[38] C. Murphy, Z. Zoomkawalla and K. Narita, Automatic test case generation and test suite reduction for closed-loop controller software, Technical Report, 2013.

[39] S. Khan, A. Nadeem, TestFilter, a Statement-coverage based test case reduction technique. Proc. 10th IEEE Int, Multitopic Conf. 2006, pp. 275-280, doi:10. 1109/INMIC.

[40] M. Weiser, Program slicing. In Proceedings of the 5th international conference on Software engineering, ICSE '81, 1981, pp. 439–449, Piscataway, NJ, USA, IEEE Pres0s.

[41] M. Weiser, Program slicing.1984. IEEE Trans, Softw. Eng. Vol. 10, No. 4, pp. 352–357.

[42] M. David Weiser, Program slices: formal, psychological, and practical investigations of an automatic program abstraction method. PhD thesis, Ann Arbor, MI, USA, 1979. AAI8007856.

[43] Bogdan Korel and Janusz Laski. Dynamic program slicing. In Information Processing Letters, 1988.

[44] Bogdan Korel and Janusz Laski. Dynamic slicing of computer programs. J. Syst. Softw., 13(3):187–195, December 1990.

[45] T. Gyim´othy, A. Besz´edes, and I. Forg´acs, An efficient relevant slicing ´ method for debugging. SIGSOFT Softw. Eng. Notes, Vol. 24, No. 6, 1999, pp. 303–321.

[46] D. Binkley, Semantics Guided Regression Test Cost Reduction. IEEE International Conference on Software Maintenance, Vol. 23, No. 8, 1997, pp. 498516.

[47] A. Mohammadian , B. Arasteh, Using program slicing technique to reduce the cost of software testing. Journal of Artificial Intelligence in Electrical Engineering, Vol. 2, No.7, 2013, pp.24-33.

[48] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms. MIT Press, Cambridge, MA

[49] Ana Emília V. B. Coutinho, Emanuela G. Cartaxo1, Patrícia D. L. Machado. "Test suite reduction based on similarity of test cases." 7st Brazilian workshop on systematic and automated software testing—CBSoft 2013.

[50] S. Xu, H. Miao and H.Gao, Test suite reduction using weighted set covering techniques, 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. IEEE, 2012.

[51] L. Zhang, D. Marinov, L. Zhang and S. Khurshid, An empirical study of JUnit test-ruite Reduction, 22nd IEEE International Symposium on Software Reliability Engineering, 2011, pp.170-179.

[52] B. Suri, I. Mangal, and V. Srivastava, Regression test suite reduction using an hybrid technique based on BCO and genetic algorithm, Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT) : 2006, 2231–5292, Vol.- II, No-1, 2

[53] Sampath, S.; Bryce, R.; Memon, A.M, "A Uniform representation of hybrid criteria for regression testing", Software Engineering, IEEE Transactions on. Vol. 39, No. 10. , 2013, pp. 1326 – 1344.

[54] S. Yoo, M. Harman, “Using hybrid algorithm for Pareto efficient multi-objective test suite minimization”, The Journal of Systems and Software .83, 2010, pp. 689–70.

[55] Nguyen Huu Phat. 2013. Slicing-based test case generation 2013, University of Bordeaux. Internship Report