CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

This section reviews the origin of testing and regression testing, the recent approaches for test suite selection, test suite minimization, test suite reduction, test suite optimization and test suite prioritization since these are related by a common focus of optimization and prioritization.

All land mark papers are systematically classified in 4 distinct categories; namely, general test case prioritization methods, feature selection in test case reduction, Particle swarm optimization in test case optimization and genetic algorithms in test case prioritization.

2.2 ORIGIN OF TESTING

Until 1956 there was no clear distinction between testing and debugging (Gelperin and Hetzel 1988) and it was the debugging oriented period. Between 1957-1978 was the demonstration oriented period, testing gets a role to play. In 1979, the separation of testing and debugging (Glenford J Myers 1979) was introduced. The destruction period was between 1979-1982, where finding the errors was the main goal. 1983-1987 was the evaluation oriented period where, a product evaluation was done to measure the quality during the software lifecycle. From 1988 onwards, it was seen as prevention oriented period where much importance was given to testing and
the main concentration was on satisfying the specification of the software so as to detect faults and to prevent faults.

2.3 ORIGIN OF REGRESSION TESTING

Regression testing is needed between two different versions of a system whenever there are some modifications in the system in order to check that the newly introduced features of the system do not interfere with the existing features. Regression Testing has been solved and explored by researchers using many ways and many techniques were proposed.

2.4 GENERAL TEST CASE PRIORITIZATION METHODS

Test case prioritization techniques are playing a vital role in testing (Harrold 2001) as researches have shown that half of the total cost of software development consists of testing activities. Research in regression testing spans a wide range of topics.

In testing, usually the program is tested with a set of test data known as test cases to uncover the errors. The same test cases may be used for running a program again after changes in the program due to some cause like change in requirement, change in technology etc., is known as regression testing. It has been tested experimentally (Coley 2007) that some of the biggest causes for project failures are lack of user input and changing or incomplete requirements. Software engineers save the test cases and re-run the test cases as regression test in later versions.

Different environments can assist regression testing particularly automation of test case execution in the regression-testing phase. The main disadvantage of regression testing is that the additional cost, time, manpower etc. that are needed for testing the program again for defects. But the
additional cost, time, manpower etc., can be reduced to some extent by not executing the full set of test cases again. So the techniques like test case reduction, test case optimization and test case prioritization (Bates and Horwitz 1993, Binkley 1995, Elbaum et al 2000, Malishevsky et al 2006, Rothermel 1999, Tonella 2006, Walcott 2006, Yau and Kishimoto 1987) may be used.

Numerous prioritization techniques have been described in the research literature (Elbaum et al 2001,2002, Jones and Harrold, 2001, Rothermel et al 2001 and Wong et al 1997). Studies (Elbaum et al 2002, Rothermel et al 1999, 2001) have shown that at least some of these techniques can significantly increase the rate of fault detection of test suites in comparison to the rates achieved by unordered or randomly ordered test suites. These early indications of potential are encouraging, however, studies have also shown that the rates of fault detection produced by prioritization techniques can vary significantly with several factors related to program attributes, change attributes, and test suite characteristics (Elbaum et al 2001, 2003).

In several instances, techniques have not performed as expected. In the empirical studies (Elbaum et al 2002), however, it was often observed that the results are contrary to this expectation. It is possible that engineers choosing to prioritize for both coverage and change attributes may actually achieve poorer rates of fault detection than if they prioritized just for coverage, or did not prioritize at all.

Test suite management and maintenance (Rothermel et al 2002, Malishevsky et al 2006) is addressed by many researchers. Reusing test cases in regression testing process is pervasive in the software industry (Rothermel et al 1999) and can save as much as one-half of the cost of software maintenance (Jefferson Offutt 1995). However, executing a set of test cases in
an existing test suite consumes a significant amount of time. There are many techniques available for test case prioritization. Different techniques (Rothermel 1999 and Elbaum 2002) like no prioritization, random prioritization, optimal prioritization, total statement coverage, additional statement coverage, total branch coverage, additional branch coverage, total fault-exposing-potential and additional fault-exposing-potential were proposed for prioritizing test cases for regression testing. The relative effectiveness of various techniques can vary significantly across target programs. The potential goal of prioritization is to increase a test suite’s rate of fault detection at the earlier stage of the software process.

Test Case Prioritization (TCP) has been primarily applied to improve the effect of regression testing (Malishevsky 2006). Test case prioritization techniques (Wong 1997) schedule test cases for regression testing in an order that attempts to maximize some objective function. It has been tested experimentally (Jeffrey and Gupta 2006) to optimize the time and cost spent on testing, which makes the test case prioritization beneficial. Regression TCP techniques (Kim and Porter 2002) used structural criteria to select the test cases.

In many applications, test cases prioritization is done based on the costs and fault metrics (Sebastian Elbaum 2001). To make the prioritization cost effective, it is very essential to select the appropriate prioritization technique based on the testing scenarios. Various metrics of programs, modifications and test suites help in identifying the technique most likely to succeed. Because of the high complexity of the factors affecting prioritization success, their interaction is also complex (Elbaum et al 2001).

Test cases were also prioritized according to the criterion of the increasing cost per additional coverage (Wong et al 1997). An implied objective of this ranking is to reveal faults earlier in the testing process. Here
the prioritization is done only for the subset of test cases selected by a safe regression test selection technique (Ball 1998, Chen 1994, Rothermel and Harrold 1997, Vokolos and Frankl 1997) from the test suite used for the program with the main aim of prioritizing the test cases for execution on a specific modified version of a program. The main disadvantage of this prioritization technique is that no mechanism is proposed for prioritizing the remaining test cases after the full coverage is attained.

Several case studies (Srikanth and Williams 2005) demonstrate the benefits of code coverage-based TCP techniques. Code coverage based TCP techniques (Kim and Porter 2002) involve ranking test cases based on the code coverage. For prioritized statement coverage, test cases are ranked based on the number of statements covered by the test case. Test suite are also prioritized based on modified condition or decision coverage (James A Jones et al 2003), branch coverage and function coverage (Onoma and Tsai 1998, Rothermel et al 1999). Using these methods test cases are prioritized based on their number of decisions or branches or functions covered by test case respectively.

Search algorithms (Li and Harman 2007) were used for regression test case prioritization. They include greedy algorithm, additional greedy algorithm, 2-optimal algorithm, k-optimal approach, hill climbing and genetic algorithms. Improving fault detection capability by selectively retaining test cases (Jeffrey and Gupta 2007) was used during test suite reduction. This new approach for test suite reduction attempts to select those additional test cases that are redundant with respect to the criterion.

Different types of testing, such as functional testing and structural testing require different testing criteria (Rothermel and Harrold 1993). There also can be cases where it is beneficial for the tester to consider multiple test criteria because the single most ideal test criterion is simply unobtainable.
One important criteria is the cost driver. Some other researchers (Malishevsky et al 2006 and Walcott et al 2006) also consider the execution time of the test suite. For the prioritization problem, there are some recent work on two objective formulation (Malishevsky et al 2006), that takes account of coverage and cost, and this approach uses the single objective of coverage per unit cost. However, this approach conflates the two objectives into a single objective.

When there are multiple competing and conflicting objectives, the optimization literature recommends the consideration of a Pareto optimal optimization approach (Elbaum et al 2000, Szidarovsky et al 1986). Such a Pareto optimal approach is able to take account of the need to balance the conflicting objectives, all of which the software engineer seeks to optimize. Multi objective regression test optimization (Mark Harman and Shin Yoo 2010) mainly concentrated on different costs.

Pareto efficient multi-objective test case selection (Shin Yoo and Mark Harman 2007) explains how the pareto optimality is used in the multi objective test case selection. Pareto optimality is that, given a set of alternative allocations and a set of individuals, allocation A is an improvement over allocation B only if A can make at least one person better off than B, without making the other worse off. The pareto efficient approach takes multiple objectives such as code coverage, past fault-detection history and execution cost, and constructs a group of non-dominating, equivalently optimal test case subsets. A re-formulation of the single–objective greedy algorithm, the Non-dominating Sorting Genetic Algorithm (NSGA-II), An island genetic algorithm variant of NSGA-II - vNSGA-II were the three algorithms used for solving the two and three objective test case selection problem.
Test suite minimization is another approach in test suite reduction. This minimization problem is also well known as the minimal hitting-set problem, or the minimal set-cover problem (Garey and Johnson 1979). This approach mainly emphasizes on how to remove the redundant and to construct the minimal test cases. Since this problem is NP complete many heuristics methods (Wong et al 1998, Leung and White 1989) are encouraged. The methods like greedy method, (Harrold et al 1993, Jeffrey and Gupta 2005, 2007, Chen and Lau 1998a), genetic method (Whitten 1998, Ma et al 2005, Mansour and El-Fakih 1999), and linear programming methods (Black et al 2004) are commonly applied. In hitting set algorithm (Harrold et al 1993), the degree of “essentialness” is used for categorizing the test cases. The heuristics G/GE/GRE algorithms (Chen and Lau 1998a) for minimizing the test cases were based on greedy, essential and 1-to-1 redundant strategies respectively.

Some of the other approaches used for test case minimization includes, Modeling the cost-benefits for regression testing (Malishevsky et al 2002), Measuring the impact of test case reduction on fault detection capability (Wong et al 1998, 1999, Rothermel et al 1998, 2002) and analyzing fault detection capability, especially with the branch coverage technique (Harrold et al 1993, Jeffrey and Gupta 2005, 2007).

Modification revealing (Rothermel and Harrold 1994a) was an approach for test case selection problem which is used for identifying all the fault-revealing test cases. Then, another concept called modification traversing for test case selection (Rothermal 2001) was introduced. Later, modification-aware approach for test case selection (Yoo and Harman 2010) was proposed for emphasizing the coverage of code changes.

The safe regression test selection approach was introduced (Rothermel and Harrold 1997) and according to this approach an algorithm is
considered as safe regression test selection if and only if it selects test cases for a test suite with full-modified function coverage.

Other approaches to test case selection problems utilize distinct techniques such as data flow analysis (Harrold and Soffa 1989), the graph-walk approach (Rothermel and Harrold 1993, 1997, 1994b), the modification-based technique (Chen et al 1994), the firewall approach (Leung and White 1990, White and Leung 1992, Zheng et al 2007) and so on. The strengths and weaknesses of these approaches were evaluated (Yoo and Harman 2010).

Measurement of regression testing process has also been researched extensively and many models and metrics have been proposed for it. Most of the research work in this area, however, has focused on test suite optimization.

2.5 FEATURE SELECTION IN TEST CASE REDUCTION

There are many good papers on feature selection which has a wide spread usage in many areas. It was used for classification (Guyon and Elisseeff 2003, Dash and Liu 1997). It was also used towards integrating feature selection algorithms for classification and clustering (Liu and Yu 2004, 2005). Later it was used in many applications of pattern recognition (Jain and Zongker 1997, Wei and Billings 2007).

Feature selection is considered as the most essential step of many pattern recognition and artificial intelligence problems (Zhang 2007). In feature selection the mutual information (Shannon 1948) is acting as standard measure of dependence which is used for feature selection and ranking as a filter in many fields like medicine, neuroscience, genomics and related fields, ecology, economics, etc (Ding and Peng 2005, Kwak and Choi 2002, Peng et al 2005). Mutual Information has been perfectly utilized in the
approach called mRMR (minimum Redundancy-Maximum Relevance) (Peng et al 2005) which aims at obtaining maximum classification or prediction performance with a minimal subset of variables by reducing the redundancies among the selected variables to a minimum and to maximize their relevance.

In feature selection, an approach was introduced for building efficient classifiers using weak features (Günter and bunke 2004, Oliveira et al 2006, Drauschke and forstner 2008) from a group of classifiers. Feature selection can also be used for handwritten script recognition (Oliveira et al 2003, Günter and bunke 2004, Morita et al 2003).

Even genetic algorithms were used for feature selection (Yang and Honavar 1998, Raymer et al 2000). Then multi objective genetic algorithms were used (Emmanouilidis 2000) for feature selection.

Test suite were reduced based on a self-organizing neural network architecture (Adenilso da Silva Simao et al 2006) using feature vector. A systematic approach was proposed for test case selection using feature model (Shuai Wang et al 2012) for product line. Goal oriented approach was proposed for hybrid test case reduction and prioritization (Alireza Ensan et al 2011) using feature model.

Even though feature selection finds a wide range of applications in many areas, it has a limited number of contributions to test case optimization.

2.6 PARTICLE SWARM OPTIMIZATION IN TEST CASE OPTIMIZATION

PSO (Russell Eberhart and James Kennedy 1995) is an optimization technique which optimizes the problem iteratively based on the movement and intelligence of swarms. The basic model for PSO (Reynolds
1987) has been extended in several different ways. The basic model was extended to incorporate the effects of fear. Sense of smell (Hartman and Benes 2006) was used to transmit emotion between animals, through pheromones in a free expansion gas. The alignment of birds and changes in leader among the birds (Hartman and Benes 2006) was also proposed. Attraction, alignment and avoidance are used (Hemerlijk and Hildenbrandt 2011) with the behavior of birds. Thus, the PSO optimization approach is used to predict the software fault (Lin and Cheng 2005).

Optimization is also used in many clustering methods like dynamic clustering (wang 2006), fuzzy clustering (zhi et al 2004), traveling salesman problems (Xu and Xiao 2006), knapsack problem (Goldbarg 2006), and minimum spanning trees (Zhu 2006). Combinatorial optimization was also used in path optimization (wang 2006), vehicle routing (wa 2004, Cernic 1999).

Ant colony optimization (ACO) (Yogesh Singh et al 2010) is used in regression testing. For document clustering analysis, (Xiaohui Cui et al 2006) flocking based approach was proposed. The proposed flock-clustering algorithm utilized the stochastic and heuristic principles for monitoring bird flocks or fish schools.

The Hybrid Particle Swarm Optimization (HPSO) algorithm (Arvinder Kaur et al 2011) was used for performing an efficient regression testing. Here, the HPSO is a combination of Particle Swarm Optimization (PSO) method and Genetic Algorithms (GA), to extend the search space for the solution.

Some marvelous algorithms have surfaced in recent times. Some of these algorithms include the genetic algorithm (Holland 1975), simulated annealing (Kirkpatrick et al 1983), particle swarm optimization (Parsopoulos
and Vrahatis 2002), ant colony optimization (Dorigo and Maria 1997), evolutionary algorithms (Schwefel 1995), altruism algorithm (Waibel et al 2011), artificial bee colony (Karaboga 2005), glowworm swarm optimization (Krishnanand and Ghose 2005), artificial immune systems (Talatahar 2010), bat algorithm (yang 2010), charged system search(Kaveh and Talatahar 2010), cuckoo search (Xin-she Yang and Suash Deb 2009), differential search algorithm(Civicioglu 2012), firefly algorithm(yang 2008), gravitational search algorithm (Rashedi et al 2009), intelligent water drops algorithm(Shah-Hosseini and Hamedauthor 2009), krill herd (Gandomi and Alavi 2012), magnetic optimization algorithm(Tayarani and Akbarzadeh 2008), multi swarm optimization(Parsopoulos and Vrahatis 2002), river formation dynamics(Pablo 2007), self-propelled particles (Vicsek et al 1995) and stochastic diffusion search(Bishop et al 1989).

2.7 GENETIC ALGORITHMS IN TEST CASE PRIORITIZATION

Genetic algorithm is the favorable field of many scientists, explorers and researchers. This enables the use of GA in miscellaneous areas like robotics (Cauvery and Viswanatha 2009), optimized telecommunication routing ( Krishnamoorthi and Sahaaya Arul Mary 2009), regression testing (Kamble 2010), data mining (Bergmann et al 2008), encryption and code breaking (Kennedy and Eberhart 2001).

The theory of natural evolution (Charles Darwin 1859) in the origin of species was developed. Over the years, organisms evolve on “survival of fittest” principle (Sivananandam and Deepa 2008).

The idea of genetic algorithm in the book Adaptation in natural and artificial System (Holland 1975) describes how to apply the natural evolution principle for optimization problems and first GA was built (Jancic and

The word “genetics” is derived from the Greek word “genesis” meaning “to grow” or “to become” (Jancic and Grundy 2003). Genetic algorithm (John Holland 1975) is an optimization and search technique based on the principles of genetics and natural selection. Test suite prioritization based on mutation faults (Do and Rothermel 2006) was done.

Genetic algorithm performs well in test case prioritization (Rothermel et al 2004) and was experimentally tested (Zheng Li et al 2007). GA converts possible solution of a problem in a genome or chromosome to a string like structure. Then GA operators such as (Mitchell 1998, Andrews 2006) selection, recombination (crossover) and mutation are applied to these genomes. The execution of GA starts with generation of random population of genomes or chromosomes. Then, the population is reproduced to produce the optimal solution to the problem (Mitchell 1998).

The usefulness of GA for function optimization and optimized GA parameters (De Jong 1975) was originated. GA has been applied in many optimization techniques (Goldberg 1989). Genes and bacteria for automatic test cases optimization (Baudry et al 2002) were implemented in the .NET environment using genetic algorithms. GA is used in model based test case generation (Praveen Ranjan Srivastava 2009). GA is used with diverse parameters (Maha alzabidi et al 2009) combinations, in order to mechanize the test data generation for path coverage. GA is also used in finding the critical path clusters to prioritize the test cases (Sangeeta Sabharwal et al 2011).
Evolutionary algorithms (EA) have played an important role in most of the optimization problems. Among the several evolutionary algorithms, simple genetic algorithm is the facile, flexible and can be applied to multi-objective optimization problems. Hybrid Genetic Algorithm (HGA) (Prabahar et al 2009) was proposed for obtaining the best possible number of test cases for the purpose of optimization.

2.8 SUMMARY

From the survey, it is identified that the feature selection algorithm finds a very limited application to test cases. From the test suite the relevant test cases are selected and the redundant ones are omitted. The birds flocking algorithm which is a particle swarm optimization technique was never used for test case optimization. GA is one such evolutionary algorithm. GA has emerged as a practical, robust optimization technique and search method. A GA is a search algorithm that is inspired by the way nature evolves species using natural selection of the fittest individuals and it is proved that Genetic Algorithms performed well in test case prioritization. Hence, based on the survey, an integrated multi objective test case prioritization system is developed for regression testing to increase the effectiveness of the testing system.

Feature selection, the first module of the integrated system is detailed in the next chapter.