REVIEW PAPER

A SURVEY OF GENETIC FEATURE SELECTION FOR SOFTWARE DEFECT PREDICTION

By

R. REENA * R. THIRUMALAI SELVI **

* Research Scholar, Department of Computer Science, Government Arts College, Nandanam, Chennai, India.
** Assistant Professor, Department of Computer Science, Government Arts College, Nandanam, Chennai, India.

ABSTRACT

Software defect prediction is an important research topic in the software engineering field, especially to solve the inefficiency and ineffectiveness of the existing industrial approach of software testing and reviews. The software defect prediction performance decreases significantly because the data set contains noisy attributes and class imbalance. Feature selection is generally used in machine learning when the learning task involves high-dimensional and noisy attribute datasets. In this survey, a Genetic Algorithm and a bagging technique is a research topic for Software Defect Prediction. The survey of publications on this topic leads to the conclusion that the field of genetic algorithms applications is growing fast. The authors overall aim is to provide an efficient feature selection for further development of the research.

Keywords: Software Defect Prediction, Genetic Algorithm, Feature Selection, Bagging Technique.

INTRODUCTION

In the research area of software defect prediction, there have been significant advances in Genetic Feature Selection for Software Defect Prediction. The authors goal is to develop the combination of genetic algorithm and bagging technique for improving the performance of the software defect prediction.

Feature selection is an important data preprocessing activity and has been extensively studied in the data mining and machine learning community. The main goal of feature selection is to select a subset of features that minimizes the prediction errors of classifiers.

Here, Genetic algorithm is applied to deal with the feature selection, and bagging technique is employed to deal with the class imbalance problem.

1. Objectives

The main objectives of this study are given below:

- Genetic algorithm is applied for solving the problem of faulty module prediction as well as finding the most important attribute for fault occurrence.
- The overall aim is to provide an efficient feature selection for further development of the research.
- A lot of research work has been carried out in the field of genetic algorithm, and various researchers have proposed many useful techniques for genetic algorithm and feature selection.

Genetic Algorithm employs on the basis of the following parameters:

- Fitness function.
- Time and space.
- Fault coverage.

2. Literature Review

Genetic Algorithm is a problem solving algorithm. It uses genetics as its model of problem solving. It is a search technique to find approximate solutions to the optimization and search problems. Genetic algorithm is applied for solving the problem of faulty module prediction and as well as for finding the most important attribute for fault occurrence. This section surveys the approaches of discourse annotation at the Genetic Algorithm. A lot of research work has been carried out in the field of genetic algorithm, and various researchers have proposed several useful techniques for genetic algorithm and feature selection.

2.1 Nikita Kravtsov and Maxim Buzdalov (2014): Worst-Case Execution Time Test Generation

This work presented an approach to Worst-Case Execution Time (WCET) test generation. It is executed in two stages. The first stage is to augment the source code of the tested program with counters which store the number of invocations of each method and loop. The second stage is to run a single-objective genetic algorithm, where the Fitness function is determined for each iteration by an objective selection algorithm.

This approach is different from other existing approaches in two aspects. First, it is more automated because it does not require a human to insert the counters into the source code. Second, two new objective selection algorithms, which do not need parameter tuning, are proposed and have shown better results. So they can be recommended as a default option for an implementation of the presented approach in a software tool for test generation [3].

2.2 Aditi Puri and Harshpreet Singh (2014): Finding Faulty Modules

AditiPuri and Harshpreet Singh developed a Genetic Algorithm to find critical classes and metrics that are fault prone. The Genetic Algorithm technique shows the high value of Probability of Detection (PD) i.e. 0.875 and the low value of Probability of False Alarms (PF) i.e. 0.44. The error and accuracy values are calculated and recorded as 0.294 and 0.705 respectively. It is therefore concluded that, the Genetic algorithm can be used for object-oriented systems and is useful in predicting the fault prone classes. The work can be extended by using other evolutionary algorithms for finding the most important attribute for fault prediction and finding the critical classes and metrics [4].

2.3 Mitchell Melanie (1999): Genetic Algorithm Introduction

This is an early attempt at the genetic algorithm. The features under study were as follows:

- GAs are promising methods for solving difficult technological problems, and machine learning. More generally, GAs are part of a new movement in computer science that is exploring biologically inspired approaches to computation. Advocates of this movement believe that, in order to create these kinds of computing systems, we need systems that are adaptable, massively parallel, able to deal with complexity, able to learn, and even creative—we should copy natural systems with these qualities. Natural evolution is a particularly appealing source of inspiration.
- Genetic algorithms are also promising approaches for modeling the natural systems that inspired their design. Most models using GAs are meant to be "gedanken experiments" or "idea models" (Roughgarden et al. 1996) rather than precise simulations attempting to match the real−world data. The purposes of these idea models are to make ideas precise and to test their plausibility by implementing them as computer programs (e.g., Hinton and Nowlan's model of the Baldwin effect), to understand and predict general tendencies of the natural systems (e.g., Echo), and to see how these tendencies are affected by changes in the details of the model (e.g., Collins and Jefferson's variations on Kirkpatrick's sexual selection model).
- Holland's Adaptation in Natural and Artificial Systems, in which GAs were defined, was one of the first attempts to set down a general framework for adaptation in nature and in computers. Holland's work has had considerable influence on the thinking of scientists in many fields, and it sets the stage for most of the subsequent work on GA theory. However, Holland's theory is not a complete description of the GA behavior [1].

2.4 K. Devika Rani Dhivya and C. Sunitha (2014): Genetic Algorithm (GA) for Randomized Unit Testing

This study focuses on the randomized unit testing with genetic algorithm. This study relied that, the randomized unit testing with genetic algorithm is less time efficient, slows down the speed of processing and also produces less optimized parameters. Randomized unit testing is a promising technology that has been shown to be

effective, but whose thoroughness depends on the settings of test algorithm parameters. A number of studies have shown less time efficiency, less optimization process and Test case generation using the randomized unit testing with genetic algorithm.

The optimization techniques for GA methods has tools like, a two-level genetic random testing system-Nighthawk, Pruned GA with FSS tool, FSS Learner into the genetic algorithmic level of Nighthawk, and: improved GA with Nighthawk for RUT are used, but the optimization of the testing process is not optimal and also produces less test case generation. The system does not consider the fitness value for the randomized unit testing, rather it generates the random selection of software units. Due to the iteration process, it becomes too complex and time consuming [5].

2.5 Isatou Hydara, Abu Bakar Md Sultan and Hazura Zulzalil (2014): Cross-Site Scripting Detection and Removal Based on Genetic Algorithms

This work heavily relied on the genetic algorithm. It presented a genetic algorithm-based approach for XSS detection and removal. Cross-site scripting is a major security problem for web applications. It can lead to account or website hijacking, loss of private information, and denial of service, all of which victimize the site users. This approach is an improvement based on two approaches. The first approach uses genetic algorithms to detect the reflected XSS vulnerabilities only but does not remove them. The second approach is able to detect and remove both the reflected and stored XSS vulnerabilities using the pattern matching technique, but not DOM-based XSS [6].

2.6 Poonam Saini and Sanjay Tyagi (2012): The Search-based Optimization Techniques-Genetic Algorithm and Clonal Selection Algorithm

In software testing, the generation of test data is one of the key steps, which have a great effect on the automation of software testing. Since the manual generation of the test data consumes much of the computational time, the process of Test Data Generation has been automated. Software Testing is also an optimization problem with the objective that the efforts consumed should be minimized. Therefore, the search based optimization techniques-Genetic Algorithm (GA) and Clonal Selection Algorithm (CSA) are used in this work. To generate the suitable data, methods were traversed to cover each (Figures 1 and 2) node. Test data values were selected based on fitness/affinity values of antibodies which satisfy the predicate node. Based on the predicate node condition, both algorithms were applied and the optimal test data was generated. Also, both techniques are compared with random testing to show that the test data generated by the search based techniques are better than the random testing as the number of test data generated for random testing is less optimal than GA, CSA [7].

2.7 Kriti Singh and Paramjeet Kaur (2014): Regression Testing using Genetic Algorithm

In this work, a genetic algorithm was developed for

a) Test Data Generation using GA:
Input: Randomly generated numbers (initial population act as test data) based on the target path to be covered.
Output: Test data for the target path.

i. Gen = 0
ii. While Gen < 100
iii. Do
iv. Evaluate the fitness value of each chromosome based on the objective function.
v. Use roulette wheel as selection operator, to select the individuals to enter into the mating pool.
vi. Perform two-point cross over on the individuals in the mating pool, to generate the new population.
vii. Perform bitwise Mutation on chromosomes of the new population
viii. Gen = Gen +1
ix. go to Step iii.
x. End
xi. Select the chromosome having the best fitness value as the desired result (test data for target path).

Figure 1. Data Generation using GA

b) Test Data Generation using CSA [17]:

i. Gen = 0
ii. Initialize random population A_0.
iii. Evaluate Affinity Function A_n
iv. if Gen > 100 then
v. output= test data
vi. Exit
vii. Else
viii. Clone A_n to A_n'
ix. Hyper-mutate A_n' to A_n''
x. Evaluate and Select A_n''
xi. Destroy and renew to construct a new population A_n
xii. Gen++
xiii. end if
xiv. goto Step iii.

Figure 2. Data Generation using CSA

prioritizing the test cases on the basis of fault coverage and execution time as input parameters of regression testing. Regression testing is a frequently executed maintenance process used to revalidate the modified software. Regression testing is one of the type of testing which works for finding the new software bugs and regression, where the functional and non functional areas of a existing system changes after the enhancements, configuration changes and patches. The intention of the regression testing is to ensure that the changes that have been made does not include new faults and also needs to identify that the changes impact other parts of the software or not. This work improves the effectiveness of algorithm with the help of Genetic Algorithm (GA). Total fault coverage within the time constrained environment on different examples is used to prioritize the test cases and their finite solution [8].

2.8 A.M. Sherry and Manish Saraswat (2014): Test Suites Prioritization for Regression Testing

In this work, A.M Sherry and Manish Saraswat developed a genetic algorithm on the test cases to prioritize their execution during Regression testing of the system or software. They used a fitness function to determine the efficiency of test case (a test case covers more number of modified lines is more efficient) and a test case sequence (or test suite) which has higher fitness value, had higher priority for execution during the testing. On applying genetic algorithm for a large number of time or generations, there is a higher probability for achieving an optimum solution [9].

2.9 J. Srividhya and K. Alagarsamy (2014): Modified Genetic Algorithm

J. Srividhya and K. Alagarsamy developed a modified genetic algorithm for reducing the cost of the regression testing. In this approach a cross sectional elitist selection is used to obtain the best individuals. When genetic algorithm is employed, a near optimal solution is also obtained. In genetic algorithm, populace of the chromosome is characterized by diverse codes, for example, real number, permutation, binary and so forth. Genetic operators such as selection, mutation and cross over are employed on the chromosome with a specific end goal to discover the fittest chromosome. The fitness of a chromosome is characterized by an objective function. Genetic algorithm includes the following steps:

- Generating the population,
- Evaluating the fitness function,
- Applying selection operator,
- Applying crossover operator,
- Evaluating and reproducing the chromosome. The optimal solution is searched in Modified Genetic Algorithm on the premise of desired populace which further could be supplanted with the new set of populace. Depending on the problem, the generation and initialization of test cases is carried out. The fitness function will help in selecting the suitable populace. Furthermore, the genetic operations are performed. Initially, ring crossover combines the two single populations [10].

2.10 Ravneet Kaur (2014): Multi-Objective Genetic Algorithm

This study was similar to (Kriti Singh and Paramjeet Kaur (2014)), but with an addition to multi-objective for the regression testing reduction. The multi objective genetic algorithm overcomes the short comes of the genetic algorithm. This work focused on optimization of the regression testing with multi-objective genetic algorithm which covered parameters like, simplicity and complexity for test cases for regression testing. Multi-objective optimization refers to the solution of problems with two or more objectives to be satisfied simultaneously. Often, such objectives are in conflict with each other, and are expressed in different units. Because of their nature, multi-objective optimization problems normally have not one but a set of solutions, which are called Pareto-optimal solutions or non-dominated solutions. When such solutions are represented in the objective function space, the graph produced is called the Pareto-front or the Pareto-optimal set. A general formulation of a multi-objective optimization problem consists of a number of objectives with a number of inequality, and equality constraints [11].

2.11 Kirandeep Kaur and Vinay Chopra (2014): Automatic Test Case Generation from UML Diagram

The main objective of this study are,

- To design and implement sequence diagrams for real time application using Agro UML and to represent the attributes and operations of objects involved in the corresponding tree structure.
- To design and implement Multi-objective Genetic algorithm for automatic test case generation for the above designed tree structures.
- To compare the effectiveness and performance of the designed Multi-objective Genetic algorithm with Genetic.

Evolutionary algorithms are used for automatic test case generation. Genetic algorithm is the most widely used technique for automatic testing. Due to limitation of Single objective genetic algorithm for which only one objective can be considered for evaluation of the test cases, a new technique known as Multi objective Genetic Algorithm was used for the test case generation from the UML sequence diagram [12].

2.12 Osaba and R. Carballedo (2014): Combinatorial Optimization Problems Solving

This study focuses on the influence of using initialization functions in genetic algorithms applied to combinatorial optimization problems. In this first stage of the research, the experimentation was conducted with the well-known TSP. This experimentation was carried out with three different heuristic functions. For each operator, the performance of four different GAs are compared. As final conclusion of this research, the efficiency of using heuristic initialization functions are highlighted. Anyway, the excessive use of them could decrease the exploration capacity of the GA, trapping the population in local optimums quickly. Therefore, the key is to maintain a balance between the individuals initialized by functions, and the individuals generated randomly [13].

2.13 Chayanika Sharma and Sangeeta Sabharwal (2013): Software Testing Techniques using Genetic Algorithm

Chayanika Sharma and Sangeeta Sabharwal developed a Software Testing Technique using Genetic Algorithm. The GA is also used with fuzzy as well as the neural networks in different types of testing. The overall aim of the software industry is to ensure delivery of high quality software to the end user. To ensure high quality software, it is required to test the software. Testing ensures that the software meets user specifications and requirements. However, the field of software testing has a number of underlying issues like effective generation of test cases, prioritisation of test cases, etc, which needs to be tackled. These issues demand on effort, time and cost of the testing. Different techniques and methodologies have been proposed for taking care of these issues. Use of evolutionary algorithms for automatic test generation has been an area of interest for many researchers. It is found that by using GA, the results and the performance of testing can be improved [14].

2.14 Josh Kounitz (2014): Understanding Software Test Cases

This study focuses on software test case understanding. A test case can have information that includes the test case name, goal, environment, steps to be taken, input and expected results. Well-designed test cases are the most important tools for discovering defects in software. These tools give you the ability to prevent serious defects, and even minor ones, from being shipped to the customers. Good test cases saves time and money and even may be the reputation of your organization [15].

2.15 Rijwan Khan and Mohd Amjad (2014): Automated Test Case Generation

This study focuse on the Automated Test Case Generation using Nature Inspired Meta Heuristics- Genetic Algorithm. For automation of software testing, the generation of test data is one of the key step and therefore the generation of testing data relates to the quality of the software production indirectly. They have applied the improved genetic algorithm for automatic test case generation with some experiment analysis and have shown in their experiment that, the improved genetic algorithm is superior to the basic genetic algorithm on effectiveness and efficiency of the automatic test case generation [16].

Conclusion

This work has presented a survey on genetic feature selection for software defect prediction. It would improve the performance of the software defect prediction. The authors research plan to develop a Genetic algorithm is applied to deal with the feature selection, and bagging technique is employed to deal with the class imbalance problem. Some systems can be used and reused in different types of genetic algorithm. But the combination of genetic algorithm and bagging technique makes an impressive improvement in prediction performance for most classifiers.

References

[1]. **Mitchell Melanie, (1999).** *Genetic Algorithm Introduction*. A Bradford Book, The MIT Press Cambridge, Massachusetts, London, England Fifth Printing.

[2]. **Chayanika Sharma and Sangeeta Sabharwal, (2013).** "A Survey on Software Testing Techniques using Genetic Algorithm". *IJCSI International Journal of Computer Science Issues*, Vol.10(1), No.1.

[3]. **Nikita Kravtsov and Maxim Buzdalov, (2014).** "Worst-Case Execution Time Test Generation using Genetic Algorithms with Automated Construction and Online Selection of Objectives". *20th International Conference on Soft Computing MENDEL 2014*, Brno, Czech Republic, June 25 -27.

[4]. **AditiPuri and Harshpreet Singh, (2014).** "Genetic Algorithm Based Approach for Finding Faulty Modules in Open Source Software Systems". *International Journal of Computer Science & Engineering Survey (IJCSES)*, Vol.5, No.3.

[5]. **K. Devika Rani Dhivya and C. Sunitha, (2014).** "A Review on Optimization Methodologies Used for Randomized Unit Testing". *International Journal of Advanced Research in Computer Science and Software Engineering*, Vol.4, No.6, pp.748-752.

[6]. **Isatou Hydara, Abu Bakar Md Sultan and Hazura Zulzalil, (2014).** "An Approach for Cross-Site Scripting Detection and Removal Based on Genetic Algorithms". *The Ninth International Conference on Software Engineering Advances ICSEA*.

[7]. **PoonamSaini and Sanjay Tyagi, (2014).** "Test Data Generation for Basis Path Testing using Genetic Algorithm and Clonal Selection Algorithm". *International Journal of Science and Research (IJSR)*, Vol.3, No.6.

[8]. **Kriti Singh and ParamjeetKaur, (2014).** "Efficient Test Cases of Regression Testing using Genetic Algorithm". *International Journal of Advanced Research in Computer and Communication Engineering*, Vol.3, No.7.

[9]. **A.M. Sherry and Manish Saraswat, (2014).** "Test Suites Prioritization for Regression Testing using Genetic Algorithm". *IJETCAS*, pp.14-150.

[10]. **J. Srividhya and K. Alagarsamy, (2014).** "Modified Genetic Approach for Regression Testing Cost Reduction". *International Journal of Infinite Innovations in Engineering and Technology*, Vol.1, No.1.

[11]. **Ravneet Kaur, (2014).** "Multi-Objective Genetic Algorithm For Regression Testing Reduction". *IJRET: International Journal of Research in Engineering and Technology*, Vol.3, No.1.

[12]. **Kirandeep Kaur and Vinay Chopra, (2014).** "Review of Automatic Test Case Generation from UML Diagram using Evolutionary Algorithm". *International Journal of Inventive Engineering and Sciences (IJIES)*, ISSN: 2319–9598, Vol.2, No.11.

[13]. **E. Osaba And R. Carballedo, (2014).** "On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: a first study on the TSP". *IEEE Conference on Evolving and Adaptive Intelligent Systems*.

[14]. **Chayanika Sharma and Sangeeta Sabharwal, (2013).** "A Survey on Software Testing Techniques using Genetic Algorithm". *IJCSI International Journal of Computer Science*, Vol.10, No.1.

[15]. **Josh Kounitz,** "Understanding Software Test Cases".

[16]. **Rijwan Khan and Mohd Amjad, (2014).** "Automated Test Case Generation using Nature Inspired Meta Heuristics- Genetic Algorithm: A Review Paper". *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, Vol.3, No.11.

ABOUT THE AUTHORS

R. Reena is a post graduate in Computer Science, from University of Madras, Chennai, Tamilnadu. She is currently a Research Scholar in the Department of Computer Science at Government Arts College, Chennai, Tamilnadu. Her research area is Software Engineering.

R. Thirumalaiselvi has more than 20 years of experience in various Engineering, Arts and Science Colleges. At present she is working as an Assistant Professor in the Department of Computer Science at Government Arts College, Nandanam, Chennai. Her areas of specialization are Software Engineering, and Web Engineering and has published many papers both in National and International Journals and also in imanager's Journal on Software Engineering.