Comprehensive Study on Detection in Software Using Genetic Algorithms

E. Sreedevi [1],

[1] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P. – 522302.

PremaLatha V [2],

[2] Department of Computer Science and Engineering Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P. – 522302.

Dr. Sivakumar Selvarasu[3]

[3]Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, KTR Campus, Kattankulathur, Tamil Nadu

Abstract:

Predicting software defects has become a significant area of research in the field of software engineering. By concentrating on faultaprone modules, the precise prediction of defectaprone software modules can aid in software testing efforts, lower expenses, and enhance the software testing process. Data sets on software defects are asymmetrical, with relatively few faulty modules in comparison to those without defects. The presence of noisy attributes in the dataset leads to a considerable fall in the performance of software defect prediction. In this study, we suggest combining the bagging technique with evolutionary algorithms to enhance software defect prediction performance. To address the issue of feature selection, a genetic algorithm is utilised, while the bagging technique is utilised to address the issue of class imbalance. The range of applications for genetic algorithms is expanding quickly, according to a survey of papers on the subject. The primary objective of the writers is to offer a productive feature selection for the advancement of the study.

Keywords:

Software Defect Prediction, Genetic Algorithm, Feature Selection

1 Introduction

In the fields of machine learning and data analysis, genetic feature selection is a potent method for selecting the most pertinent variables or features from a dataset. It becomes an essential instrument for improving the precision and effectiveness of defect prediction models when used in software defect prediction[10]. Feature selection in this context refers to determining the most significant features or attributes of software code that is most likely to be connected to flaws or faults.

Predicting software defects is an essential effort in software engineering that attempts to find possible problems or flaws in a software system before they become an issue in a production setting. Code complexity, code churn, code coupling, and other software metrics are only a few of the many code traits and metrics that are frequently used in traditional defect prediction models. All of these measurements might not be equally useful or instructive for defect prediction, though[11].

Herein lies the role of genetic feature selection. It draws inspiration from genetic algorithms, a class of optimisation algorithms grounded in the ideas of evolution and natural selection. Using a population of possible feature subsets, genetic feature selection techniques iteratively evolve and choose the most useful feature subsets depending on how well they perform in defect prediction tasks[12].

1.1 Objectives

The following lists the primary goals of this investigation:

- A genetic algorithm is used to identify the most significant attribute associated with fault occurrence and to solve the problem of faulty module prediction.
- The ultimate goal is to offer an effective feature selection for the research's continued advancement.
- Numerous studies have been conducted in the area of genetic algorithms, and numerous researchers have put forth helpful methods for feature selection and genetic algorithms.

  The main elements of genetic feature selection for software defect prediction are summarised as follows in brief:
- Population Initialization: A population of possible feature subsets is formed at the start of the procedure. You can think of these subsets as various feature combinations from the original dataset.
- Fitness Function: Each feature subset in the population is assessed for quality using a fitness function. The fitness function evaluates the ability of a certain group of features to predict errors in the context of software defect prediction. This can entail using various feature subsets to train and evaluate machine learning models.

- Genetic Operations: The population's feature subsets are subjected to genetic operations such as crossover, mutation, and selection. By simulating natural selection and evolution, these techniques promote the survival and propagation of feature subsets with superior predictive ability.
- Iterative Improvement: Over several generations, the algorithm repeatedly improves the population by choosing feature subsets that do well in defect prediction tests. Until a stopping requirement is satisfied or the algorithm converges to a solution, the procedure is repeated.

  Finding a subset of features that reduces false positives and false negatives in defect prediction models is the ultimate goal of genetic feature selection in software defect prediction. Software engineers can concentrate their efforts on the most important sections of their code base by using this strategy to create defect prediction systems that are more accurate and efficient by minimising the amount of unnecessary or redundant functionality[13].

2. Literature Review

Since 1990, software bug prediction has gained prominence, and as software engineering has advanced, so too has the number of research articles published in this area. The vast majority of applications have flaws[14]. These fall into one of four categories: Critical, High, Medium, or Low. They may arise from small to large problems (Dhavakumar & Gopalan, 2021). The majority of the literature now in publication focuses on developing prediction models for bug identification utilising data mining classification techniques, software quality measurements, and historical data sets. This section covers earlier research on the prediction of software bugs utilising hybrid approaches, supervised and unsupervised machine learning techniques, and neural networks. Hybrid approaches are those that combine several optimisation strategies influenced by nature with machine learning techniques[15][16]. Table 1 depicts the various authors proposed models for software defect detection by using Genetic algorithms.

Natural systems and phenomena, including ant colonies, particle swarms, fireflies, bats, hawks, and genetic development, serve as the inspiration for natureainspired algorithms. These algorithms handle difficult optimisation and prediction problems by imitating the adaptive properties and behaviour of these natural systems. These algorithms have the ability to go beyond the drawbacks of conventional methods and produce predictions that are more precise and effective in the context of software failure prediction. Numerous research works have examined the predictive power of algorithms inspired by nature in software defects. Researchers have experimented with and developed a variety of machinealearning techniques in an effort to anticipate and prevent errors in production. It is well known that the most costly stage of the software development lifecycle is software maintenance[17][18].

Organisations can assist in lowering the maintenance effort, time, and total cost of a software project by using a software defect forecasting model [19]. The diverse algorithms that have been studied stem from different discoveries and associations between certain software metrics and fault proneness [20]. This study follows the recommendations of a recent comprehensive literature review [4] and employs 4 of several ML classifiers. According to two research, models for software defect prediction based on machine learning [21].

Table 1: Various Proposed Software Defect Detection models by using Genetic Algorithms.

Sno	**Authors**	**Article Title**	**Year of Publication**	**Proposed Model**
1	A.M. Sherry and Manish Saraswat	Test Suites Prioritization for Regression Testing using Genetic Algorithm[1].	2014	Test Suites Prioritization for Regression Testing
2	Rijwan Khan and Mohd Amjad	Automated Test Case Generation using Nature Inspired Meta Heuristics Genetic Algorithm: A Review Paper[2].	2014	Automated Test Case Generation
3	Ravneet Kaur	Multi Objective Genetic Algorithm For Regression TestingaReduction [3].	2014	Objective Genetic Algorithm
4	Manjula and Florence	Deep neural network based hybrid approach for software defect prediction using software metrics[4].	2018	Genetic Algorithm (GA) for feature

				selection
5	Chondrodima, Eva, Georgiou, Harris, & Pelekis, Nikos, Yannis Theodoridis	Particle swarm optimization and RBF neural networks for public transport arrival time prediction using GTFS [5] .	2022	NNs with metaaheuristic techniques
6	Ruba Abu Khurma , Hamad Alsawalqah , Ibrahim Aljarah , Mohamed Abd Elaziz and Robertas Damaševiˇcius ,	An Enhanced Evolutionary Software Defec Prediction Method UsingaIsland Moth Flame Optimization, Mathematics[6]	2021	Island Moth Flame Optimization, Mathematics
7	TUSHAR ARORA , HARSHIT SAINI , SACHIN GARG	Nature Inspired Approaches in Software Fault Prediction[7] .	2023	NatureaInspired Approaches
8	Medhunhashini ,Ksjeen marseline	A hybrid genetic based grey wolf optimized sophisticatedasupport vector machine (ssvm) model for software defect prediction[8]	2023	Hybrid genetic based grey wolf

				optimized sophisticated support vector machine

2.1 A.M. Sherry and Manish Saraswat : Test Suites Prioritization for Regression Testing

In order to prioritise the execution of test cases during system or software regression testing, authors used a genetic algorithm. A test case sequence, or test suite, with a higher fitness value is given priority for execution during testing. Authors utilised a fitness function to measure the efficiency of the test case (a test case covering more modified lines is more efficient). There is a greater chance to obtain a solution that is almost optimal when genetic algorithm repeatedly or across a large number of generations are used[1].

2.2 Rijwan Khan and Mohd Amjad : Automated Test Case Generation

The automated test case generation utilising genetic algorithms—natureainspired meta heuristics—is the main topic of this work. The creation of test data is a crucial stage in the automation of software testing, and as such, it has an indirect bearing on the calibre of software development. They used an experiment analysis to apply the improved genetic algorithm for automatic test case generation, and their findings demonstrated that the improved genetic algorithm outperforms the basic genetic algorithm in terms of efficacy and efficiency of automated test case generation [2].

2.3 Ravneet Kaur : MultiaObjective Genetic Algorithm

Similar findings were found in (Kriti Singh and Paramjeet Kaur, 2014), albeit multiaobjective regression testing reduction was added. The shortcomings of the genetic algorithm are addressed by the multiaobjective genetic algorithm. The main goal of this work was to optimise regression testing using a multi objective evolutionary algorithm, taking into account factors like test case complexity and simplicity. When two or more objectives must be met at the same time, the problem is said to be solved by multiaobjective optimisation. Frequently, these goals contradict one another and are articulated in disparate terms. Multiobjective optimisation problems, by their very nature, typically have several solutions, sometimes known as Pareto optimal or non dominated solutions. The graph that results when such solutions are represented in the objective function space is referred to as the

Paretoaoptimal set or the Pareto front. A multiobjective optimisation issue is generally formulated as a set of objectives with a number of inequality and equality constraints [3].

2.4 Manjula and Florence: Genetic Algorithm (GA) for feature selection

Using data from the PROMISE repository and the MATLAB programme, Manjula and Florence (2018) created a defect model that uses a DNN method for classification and a Genetic method (GA) for feature selection. The model was contrasted with SVM, kanearest neighbour, and Decision Tree algorithms, among others. The results of the evaluation showed that the suggested method, which achieved an accuracy of 97%, outperformed the other methods indicated. Another model for bug prediction utilising NN parameter optimisation and a GA was proposed by Wahono et al. (2014)[4].

2.5 Chondrodima, Eva, Georgiou, Harris, & Pelekis, Nikos, Yannis Theodoridis: NNs with meta heuristic techniques

It is observed that a large number of excellent models have been applied to the analysis of bug prediction. Nevertheless, these methods still have issues with algorithmic complexity, prediction accuracy, and processing time when it comes to defect prediction. Practical constraints of Deep Learning models include the necessity for huge training datasets, the difficulty of establishing the optimal hyper parameters for each problem and dataset, and their lack of interpretability. As was already said, nature inspired methods allow deep learning parameters to be optimised and can be used as a base for machine learning optimisation algorithms in the future. Current state of theaart studies have concentrated on training NNs with metaaheuristic techniques (Chondrodima et al., 2022). Our effort intends to develop a novel hybrid strategy that can improve prediction accuracy in order to overcome the aforementioned problems. Although DNN and NIAs have previously been merged, we concentrate on an alternative method that combines NIAs and DNN based GA models for fault prediction [5].

2.6 Ruba Abu Khurma , Hamad Alsawalqah , Ibrahim Aljarah , Mohamed Abd Elaziz and Robertas Damaševiˇcius : Island Moth Flame Optimization, Mathematics

In order to improve the BMFO for handling the FS problem in the field of software defect prediction, this research suggests the island model. The name of the new variation is (IsBMFO). The island model creates a collection of islands out of the moth population and uses migration to transfer traits among the islands. This idea can regulate the algorithm's convergence and increase the variety of possible solutions. Different copies of MFO are applied independently and asynchronously on each island in IsMFO. The suggested method is assessed using three metrics: recall, precision, and Gamean in addition to the statistical rank test and convergence scales. The average recall, precision, and gmean of the classifiers NB, KNN, and SVM applied without FS, with BMFOaFS, and with IsBMFOaFS were compared in the tests. The values of the evaluation measures increased dynamically. When the classifiers were applied to the datasets without FS, the lower values from the three classifiers were obtained. When the IsBMFOaFS was used, the best outcomes were obtained. The SVM classifier performed the best in three experiments, with the NB classifier coming in second. The KNN classifier produced the lowest results. Additionally, on 71% of the datasets, the classifier SVM's convergence behaviour outperformed that of the NB and KNN[6].

2.7 TUSHAR ARORA , HARSHIT SAINI , SACHIN GARG : Nature Inspired Approaches

This study examined many natureainspired optimisation algorithms' efficacy in predicting software faults. The results of the experiments showed that these algorithms may efficiently optimise test cases for defect identification in a shorter amount of time. Test cases were optimised using genetic algorithms, particle swarm optimisation, ant colony optimisation, Harris Hawks optimisation, firefly optimisation, and BAT optimisation. Three datasets (JM1, CM1, and PC1) were used to assess the algorithms' performance. The Harris Hawks Optimisation algorithm performed the best among these algorithms in terms of computing efficiency and defect identification. The outcomes also revealed that software testing can be conducted more effectively and efficiently by utilising optimisation methods that are inspired by nature. All things considered, this study offers insightful guidance to software testers and developers on selecting the best optimisation method for software failure prediction[7].

2.8 Medhunhashini1 ,Ks jeen marseline : hybrid genetic based grey wolf optimized sophisticated support vector machine

Predicting software defects is essential to improving the quality of any software. The Sophisticated Support Vector Machine (SSVM) model proposed in this paper combines the potent qualities of Grey Wolf Optimisation and Genetic Algorithm. In order to address the representativeness issue, the dataset is examined, and relevant defect data is chosen. Using defect data from JDT and MyLyn Java Projects from the AEEEM repository, the suggested SSVM model is examined. The procedure of selecting features involves the use of a genetic algorithm. Using the defect data, a support vector machine is employed to train the model. An improved method for predicting which modules are broken or not is to use a GWO algorithm to determine the hyperparameter's optimal fitness value for γ tuning. The True Positive Rate, False Positive Rate, Confusion Matrix, and F1 Score are used to gauge the effectiveness of SSVM. Comparing the classic SVM with SVM GA, the experimental findings provide a greater accuracy rate of 75.30% and 77.30%, respectively. It was possible to overcome the dataset representative difficulty by putting out the SSVM model. Future work will address the major impact of an unresolved temporal dynamic issue on prediction model performance [8].

3. Conclusion

A comparative survey on genetic feature selection for software fault prediction has been presented in this paper. It would enhance the software fault prediction's performance. The feature selection issue is addressed by the authors' research goal to create a genetic algorithm, and the class imbalance issue is addressed by using the bagging technique. Certain systems are reusable in several kinds of genetic algorithms. For the majority of classifiers, however, the combination of the bagging technique and genetic algorithm significantly improves prediction performance.

4. References:

[1] A.M. Sherry and Manish Saraswat, (2014). "Test Suites Prioritization for Regression Testing using Genetic Algorithm". IJETCAS, pp.14a150.

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

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

[4] C. Manjula, Lilly Florence, Deep neural networkabased hybrid approach for software defect prediction using software metrics, https://doi.org/10.1007/s10586a018a1696az, 2018.

[5] Chondrodima, Eva, Georgiou, Harris, & Pelekis, Nikos (2018). Yannis Theodoridis,Particle swarm optimization and RBF neural networks for public transport arrival time prediction using GTFS data. International Journal of

Information Management Data Insights, 2(2), Article 100086 ISSN 2667a0968. 10.1016/j.jjimei.2018.100086.

[6] Ruba Abu Khurma , Hamad Alsawalqah , Ibrahim Aljarah , Mohamed Abd Elaziz and Robertas Damaševiˇcius, An Enhanced Evolutionary Software Defect Prediction Method Using Island Moth Flame Optimization, Mathematics 2011, 9, 1722,2011,2a20.

[7] Tushar arora , harshit saini , sachin garg, NatureaInspired Approaches in Software Fault Prediction, IRE Journals, Volume 6, Issue 12,2019, 71a76.

[8] Medhunhashini1 ,Ks jeen marseline ,a hybrid genetic based grey wolf optimized sophisticated support vector machine (ssvm) model for software defect prediction, Journal of Theoretical and Applied Information Technology, Volume 101,No12,2014,6034a6044.

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

[10] L.Granja& M. Jino: "Technique for Regression testing: Selecting test case sets tailored to possibly modified function, In proceedings of third European conference, pagea2a11, 1999.

[11] 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.

[12] Manchala, P., & Bisi, M. (2013). Diversity based imbalance learning approach for software fault prediction using machine learning models. Applied Soft Computing, 124, 109069.

[13] Saravanan, P., & Sangeetha, V. (2015). African buffalo optimized multinomial softmax regression based convolutional deep neural network for software fault prediction. Materials Today: Proceedings, 61, 619a626.

[14] Jagtap, M., Katragadda, P., & Satelkar, P. (2017, January). Software Reliability: Development of Software Defect Prediction Models Using Advanced Techniques. In 2017 Annual Reliability and Maintainability Symposium (RAMS) (pp. 1a7). IEEE.

[15] Khatri, Y., & Singh, S. K. (2018). An effective software crossaproject fault prediction model for quality improvement. Science of Computer Programming, 226, 102918.

[16] Pushphavathi, T. P. (2017, August). An approach for software defect prediction by combined soft computing. In 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) (pp. 3003a3006). IEEE.

[17] Ma, Y., Mockus, A., Zaretzki, R., Bradley, R., & Bichescu, B. (2019). A methodology for analyzing uptake of software technologies among developers. IEEE Transactions on Software Engineering, 48(2), 485a501.