The Impact of Optimization on Software Reliability in Real-World Applications

Bhawna Kaushik[1], Suman Rani Yadav[2] and Pramila Chandel[3]
[1]*Assistant Professor, Lloyd institute of Engineering and Technology, Bhawna.kaushik@liet.in*
[2]*Assistant Professor, Lloyd institute of Engineering and Technology, Suman.yadav@liet.in*
[3]*Assistant Professor, Lloyd institute of Engineering and Technology, Pramila.chandel@liet.in*

Abstract

In technical parlance, reliability is typically defined as the probability of a product or system operating correctly within a specified environment for a predetermined duration. The pervasiveness of computer programs in contemporary society implies that any malfunction has significant repercussions. Crucial to the development of such software systems is the achievement of user expectations through high-quality production. Modified software reliability models are utilized to characterize distribution based on development process data. Once a strategy demonstrates robust correspondence with the data, it can ascertain true dependability and forecast future reliability. Various methodologies, including optimization and machine learning, have been proposed to enhance reliability prediction. These methods include genetic algorithms, fuzzy logic, and neural networks, which improve dependability through predictive validity and robustness. The effectiveness of these approaches is evident in their ability to enhance software quality by identifying and rectifying defects early in the development process.

Keywords: Software Reliability, Optimization Algorithms, Machine Learning, Genetic Algorithms, Neural Networks, Fuzzy Logic, Predictive Modeling

INTRODUCTION

In technical parlance, reliability is typically defined as the probability. It ensures the correct operation of a product or system within a specified environment for a predetermined duration. The pervasiveness of computer programmes in contemporary society implies that any malfunction of said programmes has repercussions for human beings. Crucial to the development of such software systems is the achievement of user expectations through the production of high-quality software systems. As an integral component of the software engineering process, developers endeavour to assess the dependability of their software by comparing its current threshold to its historical performance. As software maintains dependable performance, the frequency of system failures diminishes over time. [1]

Modified software reliability models are utilised to characterise this distribution in accordance with data derived from the software development process.

Once a strategy has demonstrated a robust correspondence with the data, it can be employed to ascertain the true dependability of the software and forecast its future dependability. The matter at hand pertained to whether software applications have evolved to the point where computer programmers are no longer capable of conducting sufficient testing to verify the program's proper operation. It is possible that these are the result of assertions implemented by various software reliability theories, or that subsequent programmed executions are interdependent. The probability of future systems depending on current ones is affected by both the type of the actions taken to implement continuation and the extent to which the project's internal structure has been affected. [2]

To address these concerns, it is necessary to find relationships or procedures that may be used to assess software products' value more accurately over a large range of possible states. Discussing the connections between the cracks. Having said that, certain implementations make all methods unreliable.

Information retrieval, parametric framework, and non-linear time series analysis are among the methodologies that have recently been studied for their potential to represent software dependability and framework. Three, four According to a number of studies, human programmers might benefit from using computer vision improvements

to help them deal with the many forms of unpredictability that exist in software system design. In dynamic contexts, where mistakes, insufficient data, or incorrect information might surface suddenly and unexpectedly, complex models help with decision-making and prediction. These AI approaches involve collecting procedures that are prone to mistake, ambiguity, and partial truth in an attempt to attain resilience, cost-effectiveness, and prediction validity. Some of the most important basic methods are genetic programming, genetic algorithms (GAs), fuzzy logic, and neural networks.

Usually, there are two parts to the dependability prediction approach. The training phase is the first part of the process, and the prediction step is the second. The prediction model is constructed during the training phase's initial stage, employing defect information associated with all software programme components and methods-level or class-level software metrics. Following this, the exact same methodology is implemented to predict susceptibility to errors in the subsequent iteration of the software. Classes are designated as fault-free or defective through the application of classification techniques that make use of metrics associated with fault data. The utilisation of fault prediction models to identify faulty classes within software results in an enhancement of software quality. Both the model efficacy and metrics are impacted by the model methodology (5). A multitude of scholars have devised and endorsed machine learning and statistical methodologies to enhance the efficacy of dependability prediction models through the utilisation of datasets, metrics, and feature reduction techniques. Enhancing software quality through the identification of defects.

SOFTWARE PREDICTION OF DEFECTS

This paper introduces software defect predictions through the utilisation of optimisation and machine learning methodologies. A methodical and critical evaluation is provided for this purpose, as illustrated in Figure 1.

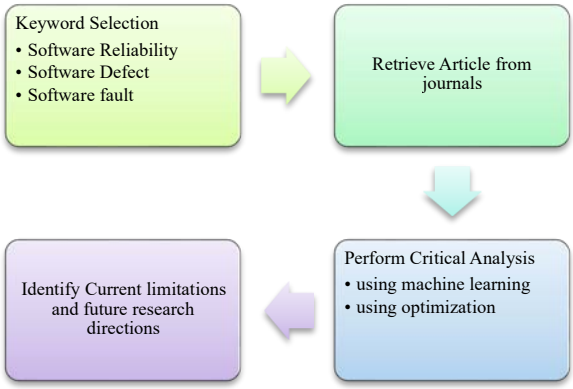

Figure 1. Approach for Systematic Meta-Analysis

Unintentionally, programmers introduce defects into software during the coding process. Feature selection stands as a highly effective approach to resolving this issue. Xiang Chen et al., motivated by the concept of IT-based engineering, transformed the challenge into a multi-objective optimisation problem. They introduced a novel technique called MOFES and utilised the PROMISE dataset, which was derived from real projects, to compare MOFES to several conventional baseline approaches. The ultimate results indicate that the method selects a reduced number of features and achieves superior prediction performance in significant implementations, all at a cost that is both practical and economical. Performance Forecasting Multiple techniques for PROMISE datasets are contrasted graphically.

For the accurate prediction of software dependability, ensemble methods were developed by Kiran et al. [2]. The provided ensembles comprise an assortment of intelligent and statistical methods (TreeNet, dynamic evolving neuro-fuzzy inference system, and backpropagation trained neural network, in addition to multivariate adaptive regression splines). Three ensembles, one of which was nonlinear, were constructed and evaluated. According to studies utilising software reliability data collected from the literature, the non-linear ensemble exhibits superior performance compared to all preceding ensembles, as well as the individual statistical and intelligent approaches.

According to what Cong et al. [3] indicated, a hybrid IEDA-SVR model should be used. In order to maintain genetic variety, it is crucial to use the chaotic mutation IEDA-SVR to forecast software reliability. The experiments made use of two real-world datasets that documented software failures. The suggested model was

tested against other models to see how well it made predictions. Applying IEDA-SVR to forecast software dependability produces outstanding results, as shown by the empirical data. In addition, IEDASVR stands out from the other comparative techniques with its superior prediction performance and relatively accurate prediction abilities. Preserving population variety may also improve the prediction model's effectiveness, according to the research. Among the two methods (EDA-SVR and the Norman and Kalman filter), the mean square value (0.2011) is the least, compared to the mean square absolute error (0.0848) and the IEDA-SVR R value (0.918).

A multi-delayed-input-based, developing connectionist technique is used to define an online, flexible software reliability prediction framework.According to four studies, the suggested approach outperforms the original NN model in terms of predictions over a broad range of computer projects when it comes to cumulative failure time prediction. The forecast has a maximum accuracy of 95%. To optimise the amount of neurons in the hidden layer of the NN architecture and delayed input neurons, a genetic algorithm is used in conjunction with known software failure time data.

In order to address software effort prediction (SEP) and software test prediction, Kassaymeh et al.[5] combined the SALP swarm method (SSA) with a back-propagation NN. Among all forecasting methods, BPNN is the most popular. The efficacy of BPNN is greatly affected by changes to model parameters, such as bias and weights. In every sample, the results show that SSA-BPNN is superior than BPNN. The following metrics are available for BPNN: R2=0.996, MAE=0.0360, RMSE=0.1907, RAE=2.4, and RRSE= 9.7 percent.

Zhen et al.[6] used a combination of WPA and PSO to forecast software reliability models. Five different kinds of industry data were used to forecast GO model values and provide forecasts. Every algorithm iterates 500 times before selecting the optimal result after running 20 times. After running the algorithms 20 times, WPA- PSO outperforms the other four datasets in terms of error rate.

An artificial neural network (ANN) model that was trained using a novel particle swarm optimisation (PSO) approach was presented by Roy et al. [7] to enhance software dependability forecasts. The proposed ANN is grounded on the phenomenon of fault creation that happens in software testing at different fault complexity levels. There are three distinct kinds of software defects that the proposed method accounts for. A neighborhood-based fuzzy PSO approach was designed to competently train the proposed ANN using software failure data. This study compares the fitting and prediction capabilities of two neural network models: one based on standard PSO and the other on neighbourhood fuzzy PSO. According to the findings, the methodology enables software to be released more rapidly. Uncertainty in prediction was accommodated by the model.

Making sure everyone involved is happy is the top priority for software developers. Delivering a high-quality software product is what it means to satisfy both producers and consumers. In order to forecast the amount of satisfaction (Q) among stakeholders, Gheisari et al.[8] introduced a new optimal mathematical model. Optimal models verify the actual data using the relationship implications of several quality factors. Constraint equations are used. The ideal model finds the highest and lowest possible values for Q. One facet of software quality is constraints. It proves that the provided outcome is optimal. The outcome proves that the value of Q drops as the value of any software quality characteristic rises. It proves that the given result is optimal.

In their proposal for software dependability, Sedlacek et al. provide a new paradigm.[9] There are a lot of software reliability models out there, however most of them aren't suitable for study of system components, such computing significance measurements. Consequently, a fresh model was unveiled. The next step is using this model to construct a syntax tree from the source code. A syntax tree is a non-language specific hierarchical representation of source code. After that, it's used to build a reliability model—a fault tree—by transforming it into a structural function. The suggested method's main advantage is that it may be used with logic differential calculus and other common approaches for system evaluation. The design does have some flaws, however, and one of them is its high dimensionality. The syntactic model yielded a dependability of 0.7567, whereas the probability calculation yielded an unreliability of 0.245.

In order to deal with nonlinear optimisation methods, Tahere et al.[10] suggest the modified differential evolution (MDE) approach. Establishing the limits of an NHPP-SR model by maximum likelihood estimation (MLE) is the issue at hand. There are two changes to DE: first, a mutation method that increases the algorithm's exploration capacity by creating a new linear mixture of multiple at least three points; second, a uniform scaling crossover technique that increases the algorithm's exploitation potential. Using five software reliability models and three software failure datasets, the efficacy of the proposed strategy is experimentally verified. The efficacy of the MDE was confirmed on a set of fifteen test cases, and the quantitative outcomes were contrasted with those of the basic DE and two other similar methods. The study data shows that the proposed method enhanced the

convergence speed by 53% compared to the basic DE. For test suites similar to those used in this study, the proposed technique similarly achieved an accurate AR. When compared to Laplace DE and RGA, the MDE showed a 57% improvement and a 43% improvement, respectively, and generated suitable ARs. Sensitivity analysis studies proved that the results were solid.

Dhavakumar et al. [11] suggest the CGWO heuristic as an innovative approach to measure SRGM properties.the eleventh The proposed method circumvents the limitations of many already-in-use approaches. The results of the evaluation criteria were derived from datasets using the parameter estimation method. The findings show that the proposed method has a reasonable capacity for forecasting, uses data attributes instead of assumptions, decreases prediction error, and can automate the whole process with no input from the user. When looking at data from chaotic graphs, the Chebyshev graph shows a respectable convergence rate of 78%. Together, the findings showed that the CGWO ftness criteria was associated with 86% of the decision variables. No human intervention is necessary to get the desired result in the SRGM when CGWO is used.

Reduced expenses and maximised efficiency by optimising the release time prediction approach. To get the optimum release time, techniques are utilised that are based on software dependability models. To get the best possible release time cost, Prashant et al.[12] suggested using both the initial and predicted release times to finish the job. The effective cost of reliability for the programme may then be calculated using this information. Additionally, the customer will be able to make a better-informed choice when choosing efficient software because of this. It follows that this programme will get the best outcomes; we optimised the cost and determined the product's delivery time using Python.

In order to minimise the financial investment required to identify a specific quantity of defects or to optimise fault detection in the face of budget constraints, Vidhyashree et al. [13] pose the optimal test activity allocation problem. In order to develop and assess expectation conditional maximisation methodologies, two distinct data sets were employed. Subsequently, we rectified the optimal test activity distribution error in order to illustrate how distributing testing resources across various tasks could potentially augment the quantity of defects identified. The value 40.64 is hypothesised to be the optimal allocation for revealing three additional issues. With a progressive increase in velocity. The individual contributions for the three covariates are 10%, 40%, and 60%, respectively, of the total effort. The reliability of software in relation to non-homogeneous Poisson processes for lifespan distributions is examined by Kim et al.14 The hazard function was diminished by the exponential, inverse-exponential, Burr-Hatke-exponential, and kairos lifetime distributions, which are prevalent in the disciplines of software reliability, economics, and the environment, respectively. As opposed to the inverse-exponential and Burr-Hatke-exponential models, the exponential distribution model produces a reduced mean square error in the present investigation. A good-fitting model is the Burr-Hatke exponential model, which yields a predicted coefficient of determination value of 95%. An model is considered efficient if it attains an estimated coefficient of determination of 95% or greater. Software fails at the final test failure time of x27=5.529 in the NHPP model; reliability is defined as the likelihood that software fails within the interval of 5.529 and 5.529 t+1. As the mission time progresses, the dependability function exhibits a non-increasing trend, and the inverse-exponential distribution model outperforms both the Burr-Hatke-exponential and exponential distribution models.

Application of the proposed fuzzy neural technique to software reliability forecasting was carried out. Several factors, such as duration to failure, mean time to failure, and so on, may be used to anticipate dependability when employing bell labs' time to failure data. Sahu et al.[16] used a method that combined neural networks with fuzzy logic. We employed algorithm-based fuzzy approach to incorporate reliability data in this technique, and the neural network tool in MATLAB was fed the fuzzy output. In addition, we used the Levenberg Marquardt algorithm, a method for neural networks, to predict reliability. The reliability prediction model's efficacy was evaluated by calculating the average normalised RMSE error. We find that out of the four approaches shown, the fuzzy-neural approach has the best overall performance, with an error of just 0.0546.

Diwaker et al.[17] deemed AI methods such as Ant Colony Optimisation (ACO), Genetic Algorithm (GA), Artificial Neural Network (NN), Particle Swarm Optimisation (PSO), and Neural Network (NN) to be significant soft computing techniques. In order to predict dependability, their paper details the inner workings of soft computing methods and how to assess them. Furthermore, the factors that are considered while estimating and forecasting reliability are investigated. Predicting and evaluating the reliability of various pieces of medical equipment, digital technology, fluid physics, and engineering may all benefit from this research.

According to Dubey et al., a CBSS reliability estimation model was proposed using ANFIS.[18] The model included the most critical factors that affect the reliability of CBSS. Data sets were used to train a hybrid NN,

which was used in ANFIS. The rule was led by this FIS-based NN. The ability of ANFIS to learn and make decisions adaptively was crucial to this model. It was also suggested to do a Mamdani FIS evaluation. When it came to figuring out if a CBSS was consistent, the ANFIS model outperformed the FIS model. Incorporating the factors discussed in CBS into a model might be a part of future research. Every facet of the growth process, both internal and external, will be taken into account.

Tong et al.[19] suggested HEEL, which stands for chaotic time series, as a solution for SRP logistic. Using chaos identification, this approach determined if the failure data was chaotic. After that, it trained the HEEL model with an excessive number of underqualified learners. Finally, a prediction was produced by applying the data to a previously trained model. We evaluated the predictive power of SRGMs with that of data-driven models. The author concluded from the data analysis that the suggested method performed best and had the highest level of predictability. As a fitness function, this technique employs MSE.

In table 1 includes critical evaluations of these books.

Table 1. Critical Analysis of Approaches Used

Ref	Year	Method	Discussions
[1]	2017	MOFES, PROMISE	Several Techniques For PROMISE Datasets Are Compared. Performance Prediction. Reasonable Computational Cost.
[3]	2014	Hybrid	IEDA-SVR model, R2 value is 0.9179, Mean Square Error was 0.201 and mean square absolute error is 0.0848.
[4]	2005	Genetic algorithm	Next-step-predictability is maximum at 95%.
[5]	2021	SALP Swarm method (SSA), Software Effort Prediction (SEP), BPNN	BPNN has 0.99531 R2, MAE is 0.036, RMSE is 0.19069, RAE in % is 2.39 and RRSE is 9.72 %.
[6]	2020	WPA-PSO	Hybrid algorithm outperforms having accuracy, optimization performance, prediction accuracy, and algorithm stability. PSO has the lowest error rate.
[7]	2019	ANN, PSO	Faster release of software. Allowed uncertainty in prediction.
[9]	2021	Syntax Tree	The Reliability calculated is 0.7567.
[10]	2020	Modified Differential Evolution (MDE), Non-Homogeneous Poisson Process (NHPP)	Improved the convergence speed by 53%.
[11]	2021	CGWO heuristic	Does not require any customer involvement. Chebyshev graph has a decent convergence rate of 78 percent.
[12]	2019	Optimization Techniques based on software reliability models	Effective cost. Determined the product's release time.
[14]	2020	Non-homogeneous Poisson processes, exponential distribution	Lower mean square error. Coefficient of determination is 95%.
[15]	2019	NHPP	MSE is 171.1531, AIC is 280.1920, and PP is 0.0517, R2 = 0.9974.
[16]	2018	Fuzzy neural approach	Lowest error around 0.0546.
[18]	2017	CBSS, ANFIS	Determined the consistency. More efficient than the FIS model.
[19]	2017	SRP based on HEEL	RMSE and average relative error. MSE is used as the fitness function. Most effective. Good predicting and performance.
[20]	2019	Ant bee colony-Particle swarm optimization (ABCPSO)	The model removes the unwanted solution in algorithm. The model was fast in operation but leads to low accuracy.
[21]	2018	Whale Optimization	The model easily handles non-linear data. Achieved better reliability.
[22]	2020	Extreme Learning Machine (ELM)	The error rate was approx. 0.05.

[23]	2009	Genetic Algorithms (GA)	The R2 single model, average ensemble and weighted average ensemble is 0.97, 0.98 and 1 respectively.
[24]	2014	Particle Swarm Optimization	The result shows that the iterations is reduced to almost 50%.
[25]	2020	Artificial Bee Colony (ABC)	The proposed approach ABCDE algorithm calculates overall system reliability to be up to 85%.
[26]	2020	Differential Algorithm	Improved convergence speed by 53%.
[27]	2021	SRDM (Software Reliability Growth Model)	RDS and SD are 0.26 and 3.20, respectively, whereas the RSD linear regression is 0.41 and SD result is 6.55.
[28]	2021	Chaotic grey wolf algorithm	Reduces errors by even more than 70%.
[29]	2018	Soft computing Methods	PSA, Grey wolf, SALP, genetic algorithm is discussed.
[30]	2021	Particle Swarm Optimization	Fitness function is introduced to increase reliability rate at MSE.

RELIABILITY PREDICTION TECHNIQUES

Reliability Prediction has been the subject of a great deal of research. Methods that are recommended for forecasting software defects include machine learning and statistical approaches. Software reliability prediction makes use of certain basic programme properties in an effort to identify software components prone to failure before the real testing process begins. As a result, software quality goals may be met with less time and money spent. Swarm evolutionary algorithms, parameter estimation, whale optimisation, ant colonies, swarm particle optimisation algorithms, optimisation based, and hybrid wolf algorithms are just a few of the prediction techniques used in software development. Other methods include effort, privacy, quality, defect, cost, and re-usability forecasting.[40] These methods of forecasting are all in their early stages. In order to create a reliable model, researchers are conducting experiments and analyses. Building a model that software professionals may use to find broken classes or modules before testing begins is called software reliability prediction (SRP).

Typically, processes involving computer-aided design and development are the focus of machine learning. Patterns in the data may be extracted from massive databases with the help of these. In the past, programmers have used neural networks (NN) to build integration models for systems in order to forecast total change or re-usability metrics. The neural network (NN) model is taught to repeatedly classify instances according to a predefined set of criteria rather than generating formulas or rules. Use of the Multilayer Perceptron (MLP) is used for faulty class handling. To classify faults according to their many kinds, radial base approaches are used.

We apply a number of statistical methods to find a basic, simple mathematical equation that describes the operation of classification. Two methods used in statistics are logistic regression and univariate binary logistic regression. When investigating data with binary variables, both approaches work well. Bayesian inference (BI) is a model-based approach that seeks to establish a relationship between metrics and software defects and their tendency to occur.

INNOVATIONS IN THE USE OF OPTIMISATION TO PREDICT THE RELIABILITY OF SOFTWARE

In their study, Rani and Mahapatra (7) presented research that extends the exponential software reliability model to quantify various aspects, such as the temporal frequency of fault detection and the occurrence rate of fault initiation. Software longevity, which includes the quantity of labour devoted to testing and the count of software defects discovered, is fundamentally influenced by module design. When assessing software dependability models, it is critical to consider the issue of resource allocation. It is critical to ascertain the optimal method for resource distribution among the components in order to achieve the necessary degree of dependability. A novel exponential distribution reliability function is integrated into our proposed multi-objective software reliability model of testing resources in order to dynamically schedule the total anticipated cost and testing effort. Extended particle swarm optimisation (EPSO) is implemented in order to achieve optimal software dependability while minimising allocation expenses. In order to fine-tune performance, scientists conduct experiments utilising entropy functions and randomised testing resource sets. A regular phase of modular testing in which the multi-objective models were applied to modules utilising a weighted cost function and test effort metrics resulted in a 99 percent reliability rate.

As a means of ensuring software dependability through the selection of redundant software systems, Gupta et al. (41) introduced a data envelopment analysis (DEA)-based nonlinear multi-objective optimisation model. When

considering data and selection, the optimisation model that is proposed incorporates both acquire and build options. The DEA technique is employed by researchers to assess software components by analysing a substantial volume of inputs or outputs supplied by diverse group members. The total efficacy rate of each programme is ascertained by utilising the aggregated data. Consideration is given to software system delivery, dependability, component compatibility, and execution time constraints in the proposed optimisation model, which aims to minimise software system cost while maximising total purchase value. Information regarding the mandatory testing of internally developed components is also provided. A practical illustration of the proposed optimisation method's implementation in software engineering modularity is provided. To the best of their knowledge, no research has been conducted on an integrative optimizer that tackles the issue of software component selection, encompassing optimally redundant build and buy options. Owing to its execution duration of 7.647 x 10das, the programme possesses an overall dependability of 80%.

Soft computing methods are the most effective for evaluating a program's predictive power, and Kumar et al.[42] used them to investigate the program's dependability. It offers a fresh comparative study to find the best and most accurate artificial neural network, based on the idea of software dependability. To improve the accuracy of software dependability, we provide a feedforward neural network that uses backpropagation in this research. Results from comparing the proposed prediction model to the previous technique using a dataset of real software evaluation concerns show that it performs better. Findings indicate that 7.21 is the lowest MMRE for FF-NN. The upper bound for linear regression is 16.60.

In order to enhance the prediction capabilities of existing PSRGMs, Jabeen et al.[44] recommended an iterative analysis method based on error-residues called highly precise error iteration analysis (HPEIAM). SRGMs improve and adjust prediction accuracy to the required level by repeatedly computing residual errors. Using three quality criteria and two sets of real software failure data, HPEIAM's performance is assessed using a number of PSRGMs. In addition, a GA was used to compare the predicted failures of HPEIAM. Results from the first rounds demonstrate that HPEIAM enhances each PSRGM's goodness-of-fit and predictive performance. At 99.2%, the R2 is high, whereas at 48.8%, it's low. In a similar vein, following the second cycle, the RMSE value changes to 2.67, and the targeted accuracy is achieved for 56 of the data periods. The estimated accuracy goes up from 170.91 to 143.01.

Software dependability utilising diverse evolutionary algorithms is the subject of many articles published in journals such as Science Direct, IEEE Transactions, and Springer, among others (see Figure 2).[41–51[Several papers were taken into account in this analysis as per the necessity. Particle Swarm Optimisation was the basis for the greatest number of studies that assessed the software's dependability. But PSO has certain major flaws, according to simulation data from other research. Research was then transferred to ACO. We looked at this technique extensively. Their findings attain global minima with less iterations and a lower value than PSO. Consequently, when it comes to distance optimisation issues, ACO is clearly superior than PSO. Grey Wolf Optimisation is another evolutionary method that is now under investigation. From the results, we may deduce that it has reduced accuracy and sluggish convergence in the latter stages of the search, an area where very few contributions have been explored. One evolutionary optimisation approach, Genetic Algorithm, has received surprisingly little attention, as seen in Figure 2. This approach outperforms all other optimisation techniques and offers a plethora of benefits. A comparison of software dependability with current methodologies is shown in Figure 3. The software reliability of Hybrid Swarm optimization[44] is 85%, while that of Multi Objective optimisation is 80%.There is a minimal SR optimisation of 75.67 percent in the syntax tree [45].The highest software dependability achieved by a genetic algorithm is around 98.47% [46].

Figure 2. Optimization Algorithm for Software Reliability Analysis

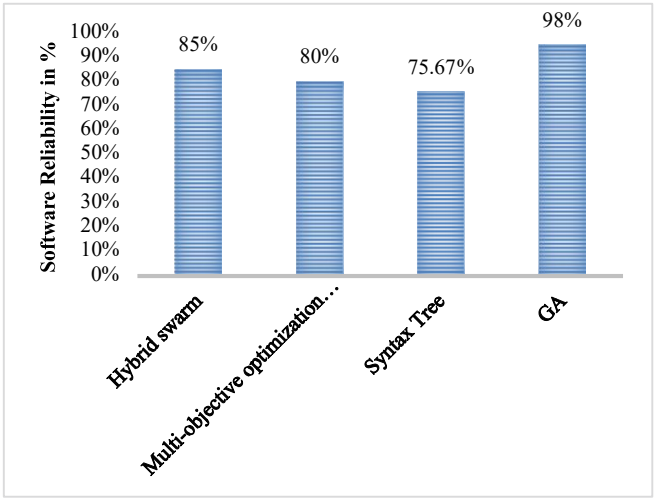

Figure 3. Reliability Performance of Optimization Algorithms

RESEARCH CHALLENGES, LIMITATIONS, AND FUTURE SCOPE

Consequently, it is essential to improve software reliability before the program's increasing relevance requires a higher level of confidence than before, and different sectors of society and industry face separate issues as a consequence. Serious consequences, including monetary losses, may result from software defects. Each year, the software business adds one billion dollars to the global economy. Bugs in software allow it to act erratically, which in turn leads to security breaches and massive financial losses. In order to check for unusual situations, reliable software needs incorporate additional code, which is often redundant. The dependability of system software is crucial due to its essential function. Therefore, it is essential to enhance the software's dependability before deploying it. Software dependability will be enhanced by the proposed endeavour. The above literature review shows that Software Reliability Optimisation has been the topic of a great deal of research.

To introduce reliability growth procedures, many methods have been used, such as:

- Relying on unfinished debugging
- The severity of the software's bugs determines
- Actions grounded in experiments
- Making Little Use of Unification Schemes
- Shifting from a one-dimensional to a two-...
- The aforementioned models have the following serious flaws:
- Determining the boundaries of failure.
- In order to find out which of the identified failure spots are statistically significant.
- The goal is to determine the parameters that provide the most accurate multi-dimensional dependability estimate.
- Locating the software's weak spots that pertain to artificial intelligence.
- We will expand on this study in the future to pinpoint the vulnerable locations that might cause an assault.

CONCLUSION

The impact of optimization on software reliability is profound, addressing the inherent unpredictability of software systems and improving their performance. Various methodologies, such as genetic algorithms, fuzzy logic, and neural networks, offer significant enhancements in predicting software reliability. These techniques have been tested on real-world datasets, demonstrating superior prediction accuracy and efficiency. For instance, the IEDA-SVR model and the hybrid WPA-PSO approach have shown remarkable results in terms of mean square error and population variety preservation. Moreover, models like SSA-BPNN and ANN with PSO optimization facilitate faster software releases and accommodate uncertainty in predictions. The adoption of these methods leads to better resource allocation, cost optimization, and improved software quality. Future research should focus

on refining these models, addressing current limitations, and exploring new optimization techniques to further enhance software reliability. The proposed study's advancements indicate that integrating optimization algorithms with software reliability models is crucial for developing robust, reliable, and high-quality software systems, which are essential in today's technologically driven society.

REFERENCES

1. X. Chen, Y. Shen, Z. Cui, X. Ju. Applying feature selection to software defect prediction using multi-objective optimization. *Proc. - Int. Comput. Softw. Appl. Conf.* **2017**, 2, 54–59.
2. N. Raj Kiran, V. Ravi. Software reliability prediction by soft computing techniques. *J. Syst. Softw.* **2008**, 81 (4), 576–583.
3. C. Jin, S.W. Jin. Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. *Appl. Soft Comput.* **2014**, 15, 113–120.
4. L. Tian, A. Noore. On-line prediction of software reliability using an evolutionary connectionist model. *J. Syst. Softw.* **2005**, 77 (2), 173–180.
5. S. Kassaymeh, S. Abdullah, M. Al-Laham, et al. Salp Swarm optimizer for modeling software reliability prediction problems. *Neural Process. Lett. 2021 536* **2021**, 53 (6), 4451–4487.
6. L. Zhen, Y. Liu, W. Dongsheng, Z. Wei. Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. *IEEE Access* **2020**, 8, 29354–29369.
7. P. Roy, G.S. Mahapatra, K.N. Dey. Forecasting of software reliability using neighborhood fuzzy particle swarm optimization based novel neural network. *IEEE/CAA J. Autom. Sin.* **2019**, 6 (6), 1365–1383.
8. M. Gheisari, D. Panwar, P. Tomar, et al. An optimization model for software quality prediction with case study analysis using MATLAB. *IEEE Access* **2019**, 7, 85123–85138.
9. P. Sedlacek, E. Zaitseva. Software reliability model based on syntax tree. *Int. Conf. Inf. Digit. Technol. 2021, IDT 2021* **2021**, 73–82.
10. T. Yaghoobi. Parameter optimization of software reliability models using improved differential evolution algorithm. *Math. Comput. Simul.* **2020**, 177, 46–62.
11. P. Dhavakumar, N.P. Gopalan. An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, 12 (2), 3177–3188.
12. P. Prashant, A. Tickoo, S. Sharma, J. Jamil. Optimization of cost to calculate the release time in software reliability using python. *Proc. 9th Int. Conf. Cloud Comput. Data Sci. Eng. Conflu. 2019* **2019**, 470–474.
13. V. Nagaraju, C. Jayasinghe, L. Fiondella. Optimal test activity allocation for covariate software reliability and security models. *J. Syst. Softw.* **2020**, 168, 110643.
14. H.C. Kim. A study on comparative evaluation of software reliability model applying modified exponential distribution. *Int. J. Eng. Res. Technol.* **2020**, 13 (5), 867–872.
15. Q. Li, H. Pham. A generalized software reliability growth model with consideration of the uncertainty of operating environments. *IEEE Access* **2019**, 7, 84253–84267.
16. K. Sahu, R.K. Srivastava. Soft computing approach for prediction of software reliability. *ICIC Express Lett.* **2018**, 12 (12), 1213–1222.
17. C. Diwaker, P. Tomar, R.C. Poonia, V. Singh. Prediction of software reliability using bio inspired soft computing techniques. *J. Med. Syst. 2018 425* **2018**, 42 (5), 1–16.
18. S.K. Dubey, B. Jasra. Reliability assessment of component-based software systems using fuzzy and ANFIS techniques. *Int. J. Syst. Assur. Eng. Manag.* **2017**, 8 (2), 1319–1326.
19. H. Tong, B. Liu, Y. Wu, B. Xu. Software reliability prediction using chaos theory and heterogeneous ensemble learning. *Risk, Reliability and Safety: Innovating Theory and Practice - Proceedings of the 26th European Safety and Reliability Conference, ESREL 2016,* **2017**, 389.
20. Z. Li, M. Yu, D. Wang, H. Wei. Using hybrid algorithm to estimate and predicate based on software reliability model. *IEEE Access* **2019**, 7, 84268–84283.
21. K. Lu, Z. Ma. Parameter Estimation of software reliability growth models by a modified whale optimization algorithm. *Proc. - 2018 17th Int. Symp. Distrib. Comput. Appl. Bus. Eng. Sci. DCABES 2018* **2018**, 268–271.
22. P.R. Bal, S. Kumar. WR-ELM: Weighted regularization extreme learning machine for imbalance learning in software fault prediction. *IEEE Trans. Reliab.* **2020**, 69 (4), 1355–1375.
23. S.H. Aljahdali, M.E. El-Telbany. Software reliability prediction using multi-objective genetic algorithm. *2009 IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA 2009* **2009**, 293–300.
24. K. Bidhan, A. Awasthi. Estimation of reliability parameters of software growth models using a variation of Particle Swarm Optimization. *Proc. 5th Int. Conf. Conflu. 2014 Next Gener. Inf. Technol. Summit* **2014**, 800–805.
25. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarmevolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.

26. T. Yaghoobi. Parameter optimization of software reliability models using improved differential evolution algorithm. *Math. Comput. Simul.* **2020**, 177, 46–62.
27. P. Kumar, S.K. Singh, S.D. Choudhary. Reliability prediction analysis of aspect-oriented application using soft computing techniques. *Mater. Today Proc.* **2021**, 45, 2660–2665.
28. P. Dhavakumar, N.P. Gopalan. An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm. *J. Ambient Intell. Humaniz. Comput.* **2021**, 12 (2), 3177–3188.
10. C. Diwaker, P. Tomar, R.C. Poonia, V. Singh. Prediction of Software Reliability using Bio Inspired Soft Computing Techniques. *J. Med. Syst.* **2018**, 42 (5), 1–16.
29. L. Yang, Z. Li, D. Wang, H. Miao, Z. Wang. Software defects prediction based on hybrid particle swarm optimization and sparrow search algorithm. *IEEE Access* **2021**, 9, 60865–60879.
30. X. Cai, S. Geng, D. Wu, J. Chen. Unified integration of many-objective optimization algorithm based on temporary offspring for software defects prediction. *Swarm Evol. Comput.* **2021**, 63, 100871.
31. S. Yadav, K. Mohan G. Parameter estimation techniques of software reliability growth models: a critical research with experimentation. *Int. J. Recent Technol. Eng.* **2019**, 8 (4), 7763–7770.
32. Jiarui Wang. Hybrid Wolf Pack and Particle Swarm Optimization Algorithm for Multihop Routing Protocol in WSN. *J. Netw. Commun. Syst.* **2020**, 3 (3), 37–44.
33. K. Lu, Z. Ma. A modified whale optimization algorithm for parameter estimation of software reliability growth models. *J. Algorithms Comput. Technol.* **2021**, 15.
34. A. Rajasekaran, R. Varalakshmi. An optimized feature selection using fuzzy mutual information based ant colony optimization for software defect prediction. *Int. J. Eng. Technol.* **2017**, 7 (1.1), 456–460.
35. H. Okamura, A. Murayama, T. Dohi. A Unified Parameter Estimation Algorithm for Discrete Software Reliability Models. *OPSEARCH 2005 424* **2017**, 42 (4), 355–377.
36. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarmevolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.
37. R.S. Wahono, N. Suryana. Combining particle swarm optimization based feature selection and bagging technique for software defect prediction. *Int. J. Softw. Eng. its Appl.* **2013**, 7 (5), 153–166.
38. R.S. Wahono, N. Suryana, S. Ahmad. Metaheuristic optimization based feature selection for software defect prediction. *J. Softw.* **2014**, 9 (5).
39. L. Zhen, Y. Liu, W. Dongsheng, Z. Wei. Parameter estimation of software reliability model and prediction based on hybrid wolf pack algorithm and particle swarm optimization. *IEEE Access* **2020**, 8, 29354–29369.
40. A. Jindal, A. Gupta, Rahul. Comparative analysis of software reliability prediction using machine learning and deep learning. *Proc. 2nd Int. Conf. Artif. Intell. Smart Energy, ICAIS 2022* **2022**, 389–394.
41. G. Jabeen, P. Luo, W. Afzal. An improved software reliability prediction model by using high precision error iterative analysis method. *Softw. Test. Verif. Reliab.* **2019**, 29 (6–7).
42. Sangeeta, K. Sharma, M. Bala. An ecological space based hybrid swarmevolutionary algorithm for software reliability model parameter estimation. *Int. J. Syst. Assur. Eng. Manag.* **2020**, 11 (1), 77–92.
43. G. Jabeen, P. Luo, W. Afzal. An improved software reliability prediction model by using high precision error iterative analysis method. *Softw. Test. Verif. Reliab.* **2019**, 29 (6–7).
44. P. Sedlacek, E. Zaitseva. Software Reliability Model based on Syntax Tree. *Int. Conf. Inf. Digit. Technol. 2021, IDT 2021* **2021**, 73–82.
45. S. Sinha, N.K. Goyal, R. Mall. Survey of combined hardware–software reliability prediction approaches from architectural and system failure viewpoint. *Int. J. Syst. Assur. Eng. Manag.* **2019**, 10 (4), 453–474.
46. D. Singh, S. Sinha, V. Thada. A novel attribute based access control model with application in IaaS cloud. *J. Integr. Sci. Technol*. **2022**, 10 (2), 79–
86..
47. P. Rani, G.S. Mahapatra. A neuro-particle swarm optimization logistic model fitting algorithm for software reliability analysis. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2019**, 233 (6), 958–971.
48. M. Yazdani, M. Babagolzadeh, N. Kazemitash, M. Saberi. Reliability estimation using an integrated support vector regression – variable neighborhood search model. *J. Ind. Inf. Integr.* **2019**, 15, 103–110.
49. J.S. Wang, S.X. Li. An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism. *Sci Rep,* **2019**, 9 (1).
50. R.R. Patil, A.U. Ruby, B.N. Chaithanya, S. Jain, K. Geetha. Review of fundamentals of Artificial Intelligence and application with medical data in healthcare. *J. Integr. Sci. Technol.* **2022**, 10 (2), 126–133.