STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–12.
Published online in International Academic Press (www.IAPress.org)

New Software Reliability Growth Model: Piratical Swarm Optimization-Based Parameter Estimation in Environments with Uncertainty and Dependent Failures

Adel S. Hussain [1], Yaseen A. Oraibi [2], Zedan Z. Mashikhin [1], Ali F. Jameel [3], Mohammad A. Tashtoush [4,3,*], Emad A. Az-Zo'Bi [5]

[1]*IT Department, Amedi Technical Institutes, University of Duhok Polytechnic, Duhok, Iraq*
[2]*Baghdad College High School, Ministry of Education, Baghdad, Iraq*
[3]*Faculty of Education and Arts, Sohar University, Sohar, Oman*
[4]*Depatrment of Basic Sciences, AL-Huson University College, AL-Balqa Applied University, AL-Salt, Jordan*
[5]*Department of Mathematics and Statistics, Faculty of Sciences, Mutah University, AL-Karak, Jordan*

Abstract In this paper our software solutions are delivered and installed in field conditions that are either identical to or comparable to development and test environments. As a result, they can also be used in a variety of settings that differ from the ones in which they were created and tested. Software dependability can be difficult to increase for a variety of reasons, including a particular environment or a flaw in the code. In this research, we offer a novel software reliability model that considers operating environment unpredictability. It has been explained the proposed model and other models of the non-homogeneous Poisson process (NHPP) is demonstrated with examples. Has been used two sets of defect data from software applications. We estimated all models' parameters by using the Cuckoo Search algorithm (CS) technique. We also conducted a simulation process to determine the good model. Through the results and their comparison with other NHPP models used, the proposed model is better than the other models and fits the data better.

Keywords NHPP; SRGM; New Process; Cuckoo Search algorithm (CS); Simulation

AMS 2010 subject classifications 60J80, 60J85, 60K10

DOI: 10.19139/soic-2310-5070-2109

1. Introduction

Software reliability growth models (SRGMs) typically assume NHPP and identical testing and operating environments with independent failures [1, 2]. Studies like this address imperfect debugging in NHPP models. Researchers [3, 4] proposed SRGMs accounting for fluctuating error counts during debugging.

Uncertain operational environments comprise a range of potential situations and circumstances, encompassing elements like the operating system, ambient settings, and hardware requirements in which users utilize the program. A model that takes into account variable operating settings and incorporates a Testing coverage function's S-shaped inflection was presented by [5, 6, 7]. Both internal and exterior elements, including voltage, regulation, programming tools, test settings, and hardware requirements, are examples of Environmental Factors (EFs). As demonstrated by [8], who took into account variables such as testing effort, abilities, and coverage, testing environments might differ. Randomness in the effort was investigated by [9] under uncertain testing and operational

*Correspondence to: Mohammad A. Tashtoush (Email: tashtoush80@bau.edu.jo). Department of Basic Sciences, Al-Huson University College, Al-Baqla Applied University. Al-Salt, Jordan

ISSN 2310-5070 (online) ISSN 2311-004X (print)

conditions. Although NHPP SRGMs generally presume separate failures, software failures can happen in a dependent manner due to how EFs interact. Researchers [10, 11] considered dependent failure occurrences.

SRGMs inform release and warranty policies in addition to evaluating dependability. An ideal release strategy that takes into account poor debugging was presented by [12]. Release rules based on change-point models were presented by [13, 14]. Research such as [15, 16, 17] combined entropy principles with pre-existing measurements to generate criteria for evaluating the goodness of fit of SRGMs.[18] provided sophisticated reliability ideas for hardware and software systems and suggested maintenance plans, whereas [19], gave multi-criteria decision-making techniques for comparing SRGMs. Software reliability analysis has been used for machine learning and deep learning methods in recent work [20, 21, 22].

The proposed model effectively addresses key challenges by explicitly incorporating environmental unpredictability, leading to more accurate and realistic reliability predictions. It also accounts for dependent failures, providing a more comprehensive understanding of failure dynamics and improving risk management [23, 24, 25]. By utilizing the Cuckoo Search algorithm for parameter estimation, the model handles complex data sets with enhanced accuracy, even when data quality is limited. Its user-friendly design simplifies parameter application, encouraging adoption among practitioners [26, 27]. Furthermore, the empirical validation of the model with real software defect data demonstrates its effectiveness and reliability in practical scenarios [28].

This paper aims to accomplish two goals: first, it presents a unique SRGM that tackles dependent failures as well as unpredictable operating situations. Although previous studies have concentrated on fault dependency or unpredictable settings, our model combines the two for a more thorough examination. Second, we evaluate our model's performance using real-world datasets. Our suggested approach outperforms models that only take dependent failures or uncertain surroundings into account, according to numerical studies, and produces more accurate failure predictions. In Section 2, we provide an overview of the fundamentals of NHPP SRGMs, present the models that are already in use, and describe the model that is suggested in this work. In Section 3, the cuckoo search (CS) algorithm's process is explained. In Section 4, the datasets and benchmarks utilized for this numerical investigation are presented. The simulation research is presented in Section 5, and the specifics of the numerical example using actual data are shown in Section 6. Lastly, the study's conclusions are presented in Section 7.

2. SRGM

2.1. Non-homogeneous Poisson Process

The NHPP, which is assumed by the majority of SRGMs, is described by the following equation

$$p\left[N\left(t\right)=y\right]=\frac{\left[m\left(t\right)\right]^{y}e^{-m\left(t\right)}}{y!},\ y=1,2,3,\ldots \tag{1}$$

It describes the total number of failures up to a specific execution time t, shown as $N(t)$ $(t>0)$. The predicted cumulative number of failures at time t is represented by the mean value function $m(t)$. As follows:

$$m(t)=\int_{0}^{t}\lambda\left(u\right)du,\ 0<\tau<\infty \tag{2}$$

With $m(t)$ [23], the NHPP-based dependability function may be stated as follows. The likelihood of no failures in the time interval $(0,t)$ is defined as the reliability function $R(t)$, which is provided by:

$$R(t)=p\left\{N(t)=0\right\}=e^{-m\left(t\right)} \tag{3}$$

Reliability $R(y/t)$ generally indicates the likelihood that there won't be any failures during the period. $[\mathrm{t},\mathrm{t}+\mathrm{y}]$ is given by:

$$R\left(\frac{y}{t}\right)=p\left\{N\left(t+y\right)-N(t)=0\right\}=e^{-\left[m(t+y)-m(t)\right]} \tag{4}$$

Equation 4 is called the SRGM or software reliability based on a non-homogeneous Poisson process (NHPP). The probability density functions as follows:

$$f(y) = \lambda(t+y)\, e^{-[m(t+y)-m(t)]} \tag{5}$$

2.2. Current Software Reliability Growth Models (SRGMs)

When we solve a differential equation, we notice that its form changes depending on the assumptions specified, and this is done by creating the mean value function $m(t)$ in NHPP SRGM is generated. This is the expression of the differential equation in the following [24]:

$$\frac{d}{dt} m(t) = b(t)\,[a(t) - m(t)] \tag{6}$$

The function $b(t)$ represents the failure detection rate for each fault, and the function $a(t)$ also represents the expected number of original failures [24].

Generally, A well-known differential equation that is industry standard describes the NHPP SRGM. The following is the derivation of the suggested model's mean value function [25], in order to take into account, the unpredictable operating circumstances taken into consideration in this study:

$$\frac{d}{dt} m(t) = \gamma\, b(t)\,[a(t) - m(t)] \tag{7}$$

This equation takes into account the unpredictable working environment by including the random variable η, which follows a generalized Gamma distribution $\gamma(\alpha, \beta)$.

2.3. Proposed Model

Many existing NHPP SRGMs operate under the assumption of identical testing and operating environments, with failures occurring independently. However, in reality, software failures can be interdependent and operational settings often differ from testing conditions. For example, an error in one code segment may trigger issues in related components, and background processes might interfere with software functionality. These scenarios result in dependent failures. In addition, creating test environments that accurately mimic all operational conditions poses challenges for testers. The operating environment encompasses various factors like operating systems (Windows, Mac, Linux), and hardware specs (CPU, GPU, RAM), alongside concurrent background processes. Our proposed model addresses both dependent errors and erratic working conditions. Numerically quantifying these habitats is complex; hence Equation 7 introduces the random variable η to represent uncertain operating conditions. We introduce NHPP reliability model that takes operational environment unpredictability into account. Additionally, the ensuing presumptions [26] suggested to be:

$$a(t) = N \,\&\, b(t) = \frac{1}{b^2} t,\ b > 0 \tag{8}$$

Where $b(t)$ is represented as the fault detection rate per fault unit of time. Equation 7 may be used to derive the mean value function $m(t)$ in the following manner [16]:

$$m(t) = \int N \left(1 - e^{-\gamma \int_0^t b(x)dx}\right) dg(\gamma) \tag{9}$$

A thorough software reliability model that incorporates the uncertainty related to the defect detection rate per unit of time in operational settings was recently proposed [24]. In this case, the random variable is described by a generalized probability density function g which has two positive parameters: α and β. The Equation 9 gives the mean value function, which may be represented as follows:

$$m(t) = N\left(1 - \frac{\beta}{\alpha + \int_0^t b(x)\,dx}\right)^{\alpha} = N\left(1 - \frac{\beta}{\alpha + \int_0^t \frac{1}{b^2} x dx}\right)^{\alpha} \tag{10}$$

By simplifying Equation 10, we obtain the proposed model:

$$m(t) = N\left(1 - \frac{\beta}{\alpha + \frac{1}{2b^2}t^2}\right)^{\alpha} \tag{11}$$

3. Cuckoo Search Algorithms (CS)

In this section, a new bio-inspired optimization algorithm, namely the Cuckoo Search (CS) algorithm is proposed in (2014) by [27, 29, 30]. It mimics the hierarchal order in the Cuckoo search and the Behaviour of the Cuckoo swarm. The Cuckoo Search algorithm is particularly effective for solving complex optimization problems due to its simplicity and efficiency. The algorithm operates based on key principles derived from the natural behaviors of cuckoos [31, 32].

Optimization can be defined as a branch of knowledge, dealing with the discovery or investigation of optimal solutions to a particular problem within a set of alternatives [33], or it can be considered one of the key quantitative tools in a decision-making network where decisions must be made to optimize one or more objectives in a specific set of Circumstances. The cuckoo i, a Levy flight is performed, [27]:

$$x_i{}^{(i+1)} = x_i{}^{i} + \beta * Levy(\lambda) \tag{12}$$

Here β is the step size that should be related to the scales. Cuckoo Search has been successfully applied to a wide range of optimization problems, including function optimization, parameter estimation, feature selection, and machine learning model tuning. Its simplicity, effectiveness, and ability to handle multimodal and non-convex optimization problems make it a popular choice for optimization tasks [34].

Nesting and Replacement: In the Cuckoo Search algorithm, each cuckoo lays its eggs in the nests of other birds. The nests represent potential solutions to the optimization problem. If a host bird discovers an egg that is not its own (i.e., a poor solution), it may abandon that nest, allowing the cuckoo to take over. This process introduces a mechanism to replace the less optimal solutions with better ones.

Selection of Best Solutions: The algorithm iteratively evaluates the quality of the nests (solutions) based on a predefined fitness function. The best solutions are retained, while poorer solutions are replaced, leading to an overall improvement in the search for optimal parameters.

3.1. Application to the Proposed Model

In the context of the proposed software reliability growth model, the Cuckoo Search algorithm is utilized for parameter estimation. The methodology involves the following steps:

Step 1. Model Formulation: The proposed model is based on a non-homogeneous Poisson process (NHPP) that accounts for dependent failures and unpredictable operating conditions. The mean value function $m(t)$ is derived analytically, which serves as the foundation for parameter estimation.

Step 2. Parameter Estimation: The Cuckoo Search algorithm is employed to estimate the parameters of the proposed model. This involves:

• Initializing a population of nests (potential parameter sets).

• Evaluating the fitness of each nest using the root mean square error (RMSE) between the observed defect data and the predicted values from the model.

• Iteratively updating the nests based on the Cuckoo Search principles, where nests are replaced or improved based on their fitness scores.

Table 1. MVF for the suggested model as well as the current NHPP SRGMs

No.	Model	$m(t)$
1	DPF1 [10]	$\frac{a}{1+\left(\frac{a}{b}\left(\frac{b+c}{c+b\,e^{bt}}\right)^{\frac{a}{b}}\right)}$
2	DPF2 [11]	$\frac{a}{1+\left(\frac{a}{b}\left(\frac{1+c}{c+\,e^{bt}}\right)^{a}\right)}$
3	DS [28]	$a\left(1-(1+bt)\,e^{-bt}\right)$
4	GO [29]	$a\left(1-e^{-bt}\right)$
5	IS [30]	$\frac{a\left(1-e^{-bt}\right)}{1+\beta\,e^{-bt}}$
6	YID [31]	$a\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha\,a\,t$
7	PNZ [24]	$\frac{a\left(1-e^{-bt}\right)\left(1-\frac{\alpha}{b}\right)+\alpha\,a\,t}{1+\beta\,e^{-bt}}$
8	PZ [32] (by Pham–Zhang)	$\frac{(c+a)\left(1-e^{-bt}\right)-\left(\frac{a\,b}{b-a}\right)\left(e^{-at}-e^{-bt}\right)}{1+\beta\,e^{-bt}}$
9	TC [33]	$N\left(1-\frac{\beta}{\beta+(a\,t)^{b}}\right)^{\alpha}$
10	VTUB [25] (VTUB model)	$N\left(1-\frac{\beta}{\beta+(a\,t)^{b}-1}\right)^{\alpha}$
11	Suggested Model (New)	$N\left(1-\frac{\beta}{\alpha+\frac{1}{2\,b^{2}}t^{2}}\right)^{\alpha}$

Step 3. Simulation and Comparison: After estimating the parameters, the model's performance is assessed through simulations. The results are compared against existing NHPP models using various goodness-of-fit metrics, such as MSE, SAE, PRR, and AIC. This comparison helps to validate the effectiveness of the proposed model in accurately predicting software reliability under uncertain conditions.

4. Comparing Models

A set of comparison standards is provided in this section. that we will use to objectively analyse the models in order to determine which one is the best.

4.1. Model Comparison Criteria

The model parameters are estimated using MATLAB software that applies the root mean square error (RMSE) method once the analytical formula for the mean function m(t) has been derived. The five standard metrics (predictive power (PP), mean square error (MSE), absolute sum error (SAE), prediction risk ratio (PRR), and Akaike information criterion (AIC)) are used to test the recommended model for goodness of fit and to compare it with other models [24]. These criteria apply to the proposed model and some common NHPP models, such as those listed in Table 1. It is important to note that models 9 in Table 1 take into account the unpredictability of the environment. The names of the authors or the attributes of each model are abbreviated. Although the proposed model (New) takes into account dependent failures and unpredictable operational circumstances, VTUB assumes uncertain operating environments.

4.2. Limitations of the Proposed Model

The proposed model, while addressing dependent failures, makes certain assumptions about these dependencies, such as linear relationships, which may not always align with real-world scenarios. Although it considers environmental unpredictability, it might oversimplify the diverse factors affecting software performance, such as varying hardware, software configurations, and user interactions. The model's accuracy is also contingent on the quality and availability of defect data; incomplete or biased data can significantly impact results. Additionally, the model may not fully account for the complexity of modern software systems, where interactions among numerous

components can lead to unpredictable behaviors. It assumes static parameters over time, potentially overlooking the dynamic nature of software development, which requires ongoing model updates. Finally, while validated with specific defect data, the model's generalizability to other software systems or domains may be limited.

4.3. Estimation of the Cuckoo Search Algorithm

In this section, we estimate the mean value function of the NHPP SRGM by using the Cuckoo Search (CS) method, taking into account all the evaluation criteria mentioned in Table 4. The suggested algorithm is defined as follows using Matlab2019a:

Step 1: Identify each of: the number of particles $N = 50$; the number of iterations with $i_max = 100$;

Step 2: The positions of each particle, representing estimations for all parameters, are randomly determined. Initially, these positions are generated from a uniform distribution within the range [0,1].

Step 3: We define the objective function based on the models defined in Table 1.

Step 4: The fitness function is set as $RMSE$, in which $RMSE = \sqrt{\frac{\sum_{i=1}^{Q} (\widehat{\gamma_i} - \gamma)^2}{Q}}$

Step 5: Generate the initial population randomly.

Step 6: Enter the main loop of the CS.

Step 7: The Cuckoo Search algorithm is based on the following equations:

Levy Flight: $s_i(t+1) = s_i(t) + aL(\lambda)(s_i(t) - s_j(t))$

Cuckoo's Nest Selection: $s_j(t+1) = s_i(t+1)$

Random walk: $s_i(t+1) = s_i(t) + aN(0,1)$

Step 8: By periodically replacing nests, the algorithm can explore the search space more effectively and potentially find better solutions.

Step 9: Perform Greedy selection.

Step 10: The estimators of parameters are adjusted based on the resultant value of the objective function RMSE.

Step 11: Steps 4 and 7 are repeated until i_max is reached.

5. Simulation

A study employing simulation techniques is conducted to evaluate the effectiveness of various estimation methods. Utilizing data generated through simulation algorithms, these methods are compared to determine the most effective approach. The algorithms for data generation are implemented in MATLAB, with the process outlined below:

Creating Random Variables: The following stages comprise the methods used to generate random variables from a given distribution function:

1. Determine the sample size n and the parameter values of the distribution;

2. Generate random observations from the distribution for the given n and parameter;

3. Calculate the parameter value using the random sample from step 2;

4. Again Steps 2 and 3 according to the specified replication number denoted by N;

Process Simulation Guidelines: The following outline the primary steps involved in designing the experiments that are to be simulated under examination.

1. Sample Size Selection: Typically, three sample sizes (20, 50, 100) are selected to demonstrate how variations in sample size n affect the estimation of model parameters. The choice of sample size significantly impacts the efficiency and accuracy of the extracted results.

2. Setting the default values: Default values are chosen for the sample size and for the parameters of each model listed in Table 1.

3. Generating random variables that follow the distribution of each model listed in Table 1, particularly for NHPP models utilizing the Monte Carlo method;

Table 2. The Simulated RMSE of each model listed in Table 1. Estimating parameters with various sample sizes and estimation techniques when a = b = 0.5; $\alpha = \beta$ = 0.6 & N = c = 0.7

No.	**Model**	**Sample size(n)**	RMSE	**Sample size(n)**	RMSE	**Sample size(n)**	RMSE
1	DPF1	20	1.5095	50	0.9547	100	0.6751
2	DPF2	20	1.0274	50	0.6498	100	0.4595
3	DS	20	1.0612	50	0.6711	100	0.4746
4	GO	20	1.2189	50	0.7709	100	0.5451
5	IS	20	0.8061	50	0.5098	100	0.3605
6	YID	20	8.1229	50	5.1374	100	3.6327
7	PNZ	20	6.7338	50	4.2588	100	3.0115
8	PZ	20	0.5791	50	0.3663	100	0.2590
9	TC	20	1.2867	50	0.8138	100	0.5754
10	VTUB	20	1.2349	50	0.7810	100	0.5522
11	suggested model	20	0.0637*	50	0.0403*	100	0.0285*

Table 3. The Simulated RMSE of each model listed in Table 1. Estimating parameters with various sample sizes and estimation techniques when a = b = 0.6; $\alpha = \beta$ = 0.5 & N = c = 0.7

No.	**Model**	**Sample size(n)**	RMSE	**Sample size(n)**	RMSE	**Sample size(n)**	RMSE
1	DPF1	20	1.5095	50	0.9547	100	0.6751
2	DPF2	20	1.0274	50	0.6498	100	0.4595
3	DS	20	1.0770	50	0.6812	100	0.4817
4	GO	20	1.2189	50	0.7709	100	0.5451
5	IS	20	0.8065	50	0.5100	100	0.3607
6	YID	20	16.1644	50	10.2233	100	7.2289
7	PNZ	20	14.2315	50	9.0008	100	6.3645
8	PZ	20	0.5712	50	0.3612	100	0.2554
9	TC	20	1.2867	50	0.8138	100	0.5754
10	VTUB	20	1.2358	50	0.7816	100	0.5526
11	suggested model	20	0.0590*	50	00373*	100	0.0264*

4. After determining the model parameter estimator by various techniques and utilizing the root mean square error $RMSE$, contrasting various approaches takes the following form: $RMSE = \sqrt{\frac{\sum_{i=1}^{Q} (\widehat{\gamma_i} - \gamma)^2}{Q}}$

5. Random generation for each model listed in Table 1.

Following the determination of the parameters' starting values, the values were methodically changed in conjunction with the sample size and evaluated by many program iterations. $n = 20$, 50, and 100 sample sizes were used, along with other parameter value combinations, including $(a = b = 0.5; \alpha = \beta = 0.6; N = c = 0.7)$ and $(a = b = 0.6; \alpha = \beta = 0.5; N = c = 0.7)$. The findings displayed in Table 2 and Table 3 demonstrate that the RMSE value of the suggested model is consistently lower than that of the other models.

6. Numerical Examples

This section compares the goodness of fit of the proposed and current models by using real data to estimate their respective criteria. First, we fit each model (mean value function) to the dataset and use the Cuckoo search algorithm (CS) to estimate each model's parameters based on the root mean square error $RMSE$ performance. Next, we compute the criterion using the estimated values of the parameters $\hat{m(t)}$, and compare the models' goodness of fit.

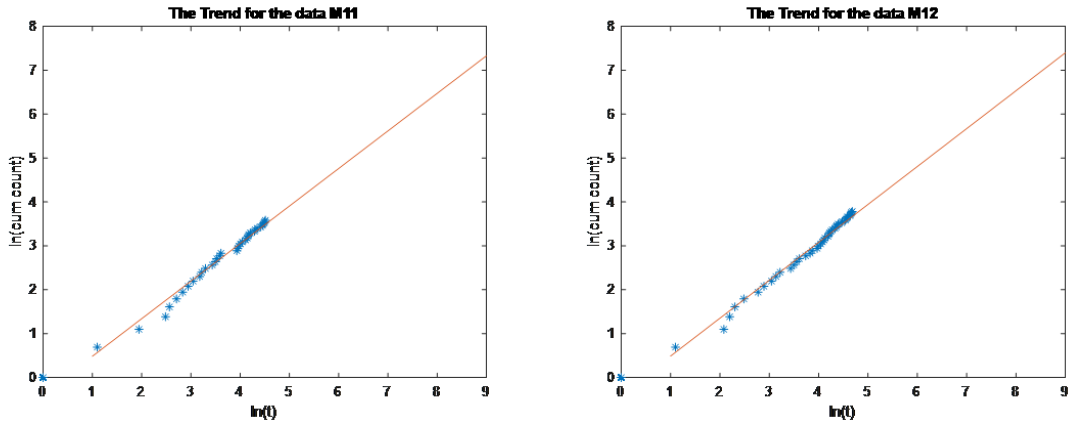

Figure 1. The cumulative data volume's logarithm is represented by the diffusive form.

6.1. Real Data

We utilized two datasets to assess the goodness of fit of various models. The first dataset $M11$ = [10 2 4 6 6 8 4 3 1 6 1 4] data used to support the findings of this study have been deposited in [23, 24], was gathered by ABC Software Company. This dataset spans 12 weeks (with time units represented in weeks), during which 55 failures were observed. It gave the second dataset command and control system developed by Bell Laboratories; this dataset contains failure data seen during system testing; 136 failures were recorded during a 23-hour period which is $M12$ = [27 16 11 10 8 1 5 3 1 4 7 5 5 6 0 5 1 1 2 1 2 1 1]. Data used to support the findings of this study have been deposited in real-time [35, 36, 37].

6.2. Goodness of Fit Tests for Data Set

The goodness-of-fit test is an essential step in statistical analysis, especially when analysing lifetime data, to find the distribution that best matches the data. Graphical techniques are frequently used in classical tests to evaluate the appropriateness of the data. This part looks at the data graphically and evaluates how well it fits the mean value function. Plotting the cumulative failures against the logarithm of time allows for this to be achieved. The data fits the function for NHPP SRGMs well if the majority of these points form a straight line. Thus, we get the following equation by taking the natural logarithm and applying it to the cumulative function of the suggested model:

$$\ln\left[m\left(t\right)\right] = \ln\left(N\right) + \alpha \ln\left(1 - \frac{\beta}{\alpha + \frac{1}{2b^2}\mathrm{t}^2}\right) \tag{13}$$

By using the programming language MATLAB, the following figure was obtained.

In Figure 1 the graphical distribution illustrates the cumulative failures of days on a logarithmic scale for the dataset under examination. It's noteworthy that the scatter plot depicts a linear relationship, suggesting the feasibility of modeling such data using a mean value function.

6.3. Criteria

Numerous standards have been put out in this study to evaluate how well a model fits the data [23]. In comparison the proposed model with ten NHPP SRGMs that already exist, nine evaluation criteria are specifically looked at. The many evaluation criteria used to assess the goodness of fit of various NHPP SRGMs in conjunction with the suggested model are compiled in Table 4, [37, 38]. These parameters measure the difference or gap between the expected number of failures as predicted by the MVF of the model, represented as $m(t_i)$, It is vital to compare the expected number of failures, represented as, $m(t_i)$, with the actual observed data, represented as y_i. In this case,

Table 4. Evaluation Criteria

No.		Criteria
1	MSE [23]	$\frac{\sum_{i=1}^{n}\left(\widehat{m}(t_i)-y_i\right)^2}{n-m}$
2	PRR [23]	$\sum_{i=1}^{n}\left(\frac{\widehat{m}(t_i)-y_i}{\widehat{m}(t_i)}\right)^2$
3	PP [23]	$\sum_{i=1}^{n}\left(\frac{\widehat{m}(t_i)-y_i}{y_i}\right)^2$
4	SAE [25]	$\sum_{i=1}^{n}\left\lvert\widehat{m}\left(t_i\right)-y_i\right\rvert$
5	R-square R^2 [28]	$1-\frac{\sum_{i=1}^{n}\left(\widehat{m}(t_i)-y_i\right)^2}{\sum_{i=1}^{n}(y_i-y_l)^2}$
6	AIC [34]	$-2\log MLF+2m$
7	PRV [30]	$\sqrt{\frac{\sum_{i=1}^{n}\left(\widehat{m}(t_i)-y_i-bias\right)^2}{n-1}}$
8	MAE [34]	$\frac{\sum_{i=1}^{n}\left\lvert\widehat{m}(t_i)-y_i\right\rvert}{n-m}$
9	MEOP [31]	$\frac{\sum_{i=1}^{n}\left\lvert\widehat{m}(t_i)-y_i\right\rvert}{n-m+1}$
10	FPA [38]	$\frac{\sum_{t=1}^{T}\text{Correct Predictions at time } t}{\sum_{t=1}^{T}\text{Total Predictions at time } t}\times 100\%$

n represents the total number of data points and m for the total number of model parameters [39, 40]. A smaller difference between the expected and actual values indicates that the MVF of the model predicts the number of failures in the dataset more accurately [10, 37, 41].

A statistical metric called the AIC evaluates a model's fit to the data. It accounts for the model's parameter count and maximizes the probability function (L) of the model. Although models with more parameters often fit data better, overfitting is avoided by the AIC, which penalizes models with too many parameters. The log-likelihood function $(logL)$ plus a penalty term determined by the number of parameters is how the AIC is computed:

$$L=\prod_{i=1}^{n}\frac{\left(\hat{m}\left(t_i\right)-\hat{m}\left(t_{i-1}\right)\right)^{y_i-y_{i-1}}}{\left(y_i-y_{i-1}\right)!} \tag{14}$$

$$logL=\sum_{i=1}^{n}\left(\left(y_i-y_{i-1}\right)-\hat{m}\left(t_{i-1}\right)-\log\left(y_i-y_{i-1}\right)!\right) \tag{15}$$

In conclusion, there are nine factors that may be used to assess a model's goodness of fit. A higher R^2 value suggests that the model fits the data better. Compared to other models using the same dataset. Are smaller values for these parameters often indicating a better model fit.

6.4. Results

Table 5 and Table 6 show the estimated parameters of the models, which were obtained using the Cuckoo Search algorithm (CS) based on the root mean square error.

Table 5. Estimation of Dataset (M11) Parameters

No.	Model	$\widehat{a}$	$\widehat{b}$	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{N}$	$\widehat{c}$
1	DPF1	9.4297	0.1140	—	—	—	10.3181
2	DPF2	3.4596	3.2228	—	—	—	6.0154
3	DS	-1.1208	0.3139	—	—	—	—
4	GO	3.1472	4.0579	—	—	—	—
5	IS	4.7331	10.6038	—	10.7170	—	—
6	YID	5.2098	6.8974	5.3214	—	—	—
7	PNZ	7.0928	10.1743	0.0354	1.7601	—	—
8	PZ	3.3667	8.5408	0.9324	1.0074	—	2.7473
9	TC	1.8582	2.3261	4.6703	5.1124	3.4883	—
10	VTUB	2.4255	1.8028	2.6347	2.7264	9.4491	—
11	NEW	—	2.6124	40.6233	0.4187	4.7557	—

Table 6. Estimation of Dataset (M12) Parameters

No.	Model	$\widehat{a}$	$\widehat{b}$	$\widehat{\alpha}$	$\widehat{\beta}$	$\widehat{N}$	$\widehat{c}$
1	DPF1	7.2508	11.6045	—	—	—	7.9885
2	DPF2	8.2236	7.6349	—	—	—	7.6659
3	DS	5.4424	2.0849	—	—	—	—
4	GO	5.9513	8.8399	—	—	—	—
5	IS	4.1066	10.0008	—	12.8173	—	—
6	YID	9.3930	8.6980	5.7867	—	—	—
7	PNZ	3.3827	3.5073	3.4221	8.0882	—	—
8	PZ	7.6529	7.1750	13.6901	6.9477	—-	6.3908
9	TC	7.0227	8.1639	13.6522	4.5633	8.2545	—
10	VTUB	4.9023	2.1408	5.9701	11.1018	6.0265	—-
11	NEW		8.8733	6.9631	9.5829	7.2516	

Estimated parameter values for the models across the two data sets will be presented in Table 7 and Table 8. It is shown that our proposed model has the lowest parameters on the dataset $M11$. Also, for the dataset, $M12$ has the lowest parameters. Moreover, these results show that our proposed model outperforms other models in estimating the total number of failures across datasets.

The MVF for each of the 11 models across the datasets $M11$ and $M12$ is shown in Figure 2.

7. Discussion

In summary, the proposed software reliability growth model offers valuable practical implications for software development practices, including enhanced reliability, risk mitigation, and improved resource management. However, organizations must also navigate challenges related to complexity, data requirements, and stakeholder resistance to fully realize the benefits of implementing the model. By addressing these challenges, the model can contribute significantly to the advancement of software reliability practices in diverse operational contexts.

Table 7. Model Criteria Value Comparison for Dataset (M11)

Model	MSE	PRR	PP	SAE	R^2	AIC	PRV	MAE	MEOP	FPA
DPF1	2.4568	81.0992	71.0598	5.4297	0.7650	38.6958	0.4177	0.4525	2.7975	27.9751
DPF2	0.0243	5.3056	6.0496	0.5404	0.5124	37.7076	0.5797	0.1450	3.2950	32.9500
DS	2.1852	5.9932	4.4978	5.1208	0.4244	39.6251	1.8887	0.4267	3.6767	36.7671
GO	0.0606	3.5202	4.6105	0.8528	0.5526	30.1451	0.4254	0.1711	3.3211	33.2110
IS	0.0448	15.1165	13.9360	0.7331	2.4840	32.1389	0.8944	0.1611	3.1889	31.8890
YID	1.9304	2.3286	2.3256	1.5220	0.8139	34.1689	43.6975	0.8316	2.5816	25.8161
PNZ	1.1574	206.5067	0.5283	3.7268	0.8679	34.2519	1.4174	0.3106	3.5606	35.6061
PZ	0.3724	29.8090	26.1530	2.1140	0.5382	29.5514	0.9988	0.1762	3.0738	30.7380
TC	0.0228	5.4831	6.1916	0.5117	0.5479	28.6368	0.5912	0.1426	3.2926	32.9261
VTUB	2.4744	81.4647	71.3873	5.4491	0.8468	28.6284	0.4053	0.4541	2.7959	27.9590
NEW	0.0220	1.1802	2.6102	1.3844	0.3474	27.1866	0.3469	0.1154	2.3654	23.6540

Table 8. Model Criteria Value Comparison for Dataset (M12)

Model	MSE	PRR	PP	SAE	R^2	AIC	PRV	MAE	MEOP	FPA
DPF1	42.2654	3.0607	2.3557	3.2215	0.2085	138.7791	169.6215	6.1393	127.8607	12.7860
DPF2	40.8883	1.4577	1.0147	3.2743	0.6596	142.8089	171.5776	4.0293	129.9707	12.9970
DS	21.1744	11.6273	8.0294	3.2854	0.6971	208.6486	171.2299	3.6643	130.4157	13.0415
GO	55.3108	1.4650	1.3883	3.2572	0.5089	117.2816	170.7210	4.7930	129.2870	12.9287
IS	16.3276	1.4784	5.4082	3.3015	0.8062	122.8768	172.5658	3.0216	131.0584	13.1058
YID	6.3703	1.5784	2.9258	2.8341	0.6521	121.8758	205.3538	21.7162	112.3638	11.2363
PNZ	6.2979	1.0324	1.5742	2.9782	0.9415	116.4497	178.7504	15.9507	118.1293	11.8129
PZ	10.1581	1.4653	2.5790	3.3242	0.9055	126.0082	173.6287	2.1113	131.9687	13.1968
TC	89.1623	2.3183	1.7229	3.2437	0.4257	114.9172	170.4178	5.2504	128.7496	12.8749
VTUB	72.3238	1.0223	1.4308	3.2554	0.4940	141.8089	170.6458	4.8647	129.2153	12.9215
NEW	6.2978	1.0165	1.0146	2.3625	0.0156	114.7360	169.6055	1.4465	112.8311	11.2311

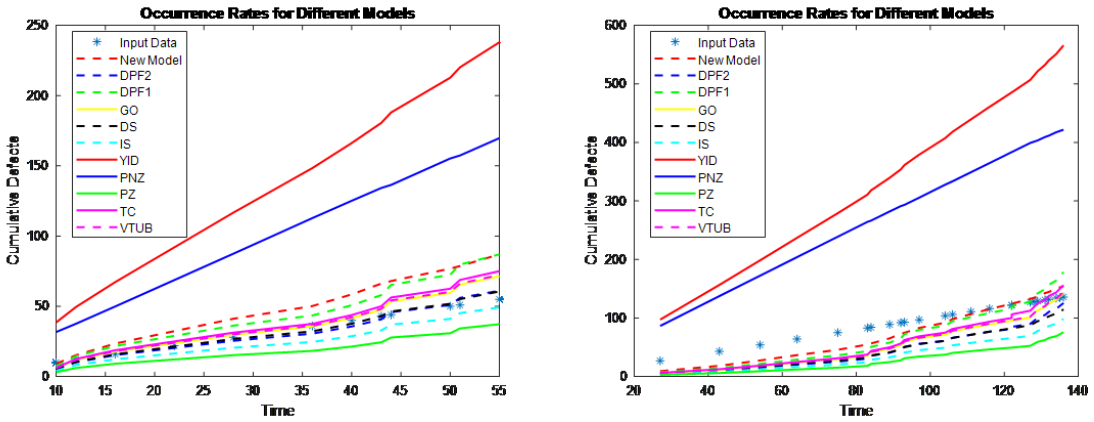

Figure 2. Average value functions for every model for datasets M11 and M12 (A) and (B).

8. Conclusions

This paper aims to propose a model that accounts for unexpected operating conditions and dependent failures. The results show that our model was better than those that considered unexpected operating conditions (VTUB) or dependent failures (DPF 1 and DPF 2). We also presented a method for evaluating the reliability of programs

using the Cuckoo search algorithm. This emphasizes the need to improve reliability. In this study, the Cuckoo search algorithm was used to achieve the objectives of early reliability evaluation. In operational contexts, pre-existing models are usually used to test data for predictions. However the difference between the operational and test settings means that the operating environment's unpredictability must be taken into account. Our new software dependability model is based on the use of RMSE, which is frequently used to simulate unpredictable operational situations. The superiority of our model was illustrated by the findings from Table 7 and Table 8, which summarized the parameters calculated using the Cuckoo search method and showed reduced values of MSE, SAE and PRR compared to other models. These results demonstrate how well our model predicts software faults and how it can be used to increase dependability.

Acknowledgement

The authors express their deep gratitude to Duhok Polytechnic University for the facilities provided, which contributed to improving the quality of this work.

REFERENCES

1. Y.S. Huang, K.C. Chiu, and W.M. Chen, *A software reliability growth model for imperfect debugging*, J. Syst. Softw, vol. 188, pp. 111267, 2022. https://doi.org/10.1016/j.jss.2022.111267
2. H. Luo, L. Xu, L. He, L. Jiang, and T. Long, *A Novel Software Reliability Growth Model Based on Generalized Imperfect Debugging NHPP Framework*, IEEE ACCESS, vol. 11, pp. 71573–71593, 2023. https://doi.org/10.1109/access.2023.3292301
3. K.C. Chiu, Y.S. Huang, and I.C. Huang, *A study of software reliability growth with imperfect debugging for time-dependent potential errors*, Int. J. Ind. Eng. Theory, Appl. Pract, vol. 26, no. 3, pp. 376-393, 2019.
4. R. Gupta, M. Jain, and A. Jain, *Software reliability growth model in a distributed environment subject to debugging time lag*, Perform. Predict. Anal. Fuzzy, Reliab. Queuing Model. Theory Appl., pp. 105-118, 2019.
5. S. K. Pradhan, A. Kumar, and V. Kumar, *A New Software Reliability Growth Model with Testing Coverage and Uncertainty of Operating Environment Comput*, Sci. Math. Forum, vol. 1, no. 0, pp. 112–126, 2023.
6. V. Pradhan, J. Dhar, and A. Kumar, *Testing coverage-based software reliability growth model considering the uncertainty of operating environment*, Syst. Eng., vol. 26, no. 4, pp. 449-462, 2023.
7. S. K. Pradhan, A. Kumar, and V. Kumar, *A Testing Coverage Based SRGM Subject to the Uncertainty of the Operating Environment*, Computer Sciences and Mathematics Forum, vol. 7, no. 1, pp. 44-53, 2023.
8. M. A. Haque and N. Ahmad, *Software reliability modeling under an uncertain testing environment*, Int. J. Model. Simul., pp. 1-7, 2023.
9. S. Chatterjee, D. Saha, A. Sharma, and Y. Verma, *Reliability and optimal release time analysis for multi-up-gradation software with imperfect debugging and varied testing coverage under the effect of random field environments*, Ann. Oper. Res., vol. 312, no. 1, pp. 65-85, 2022.
10. D. H. Lee, I. H. Chang, and H. Pham, *Software reliability model with dependent failures and SPRT*, Mathematics, vol. 8, no. 8, pp. 1366, 2020.
11. Y. S. Kim, K. Y. Song, H. Pham, and I. H. Chang, *A software reliability model with dependent failure and optimal release time*, Symmetry (Basel)., vol. 14, no. 2, pp. 343, 2022.
12. A. R. Raheem, S. Akthar, and S. M. Rafi, *An Imperfect Debugging Software Reliability Growth Model: Optimal Release Problems through Warranty Period based on Software Maintenance Cost Model*, Rev. GEINTEC-GESTAO Inov. E Tecnol., vol. 11, no. 4, pp. 4623–4631, 2021.
13. S. Ke and C. Huang, *Software reliability prediction and management: A multiple change-point model approach*, Qual. Reliab. Eng. Int., vol. 36, no. 5, pp. 1678–1707, 2020.
14. Y. Minamino, S. Inoue, and S. Yamada, *Change-point–based software reliability modeling and its application for software development management*, Recent Advancements in Software Reliability Assurance, CRC Press, pp. 59-92, 2019.
15. P. Saxena, V. Kumar, and M. Ram, *A novel CRITIC-TOPSIS approach for optimal selection of software reliability growth model (SRGM)*, Qual. Reliab. Eng. Int., vol. 38, no. 5, pp. 2501–2520, 2022.
16. V. Kumar, P. Saxena, and H. Garg, *Selection of optimal software reliability growth models using an integrated entropy–Technique for Order Preference by Similarity to an Ideal Solution (TOPSIS) approach*, Math. Methods Appl. Sci., vol. 8, no. 8, pp. 1366, 2021.
17. R. Garg, S. Raheja, and R. K. Garg, *Decision support system for optimal selection of software reliability growth models using a hybrid approach*, IEEE Trans. Reliab., vol. 71, no. 1, pp. 149-161, 2021.
18. M. Zhu, *A new framework of complex system reliability with imperfect maintenance policy*, Ann. Oper. Res., vol. 312, no. 1, pp. 553–579, 2022.
19. T. Yaghoobi, *Selection of optimal software reliability growth model using a diversity index*, Soft Comput., vol. 25, no. 7, pp. 5339–5353, 2021.
20. K. K. San, H. Washizaki, Y. Fukazawa, K. Honda, M. Taga, and A. Matsuzaki, *Deep Cross-Project Software Reliability Growth Model Using Project Similarity-Based Clustering Mathematics*, Math. Softw. Reliab. Qual. Assur., pp. 167, 2021.

21. N. Shirawia, A. Qasimi, M. Tashtoush, N. Rasheed, M. Khasawneh, E. Az-Zobi, *Assessment of the Calculus Students by Using Scoring Rubrics in Composition and Inverse Function*, Applied Mathematics and Information Sciences, vol. 18, no. 5, pp. 1037–1049, 2024.
22. Y. Wardat, M. Tashtoush, R. Alali, S. Saleh, *Artificial Intelligence in Education: Mathematics Teachers' Perspectives, Practices and Challenges*, Iraqi Journal For Computer Science and Mathematics, vol. 5, no. 1, 60-77, 2024.
23. E. Az-Zo'bi, K. Al Dawoud, M. Marashdeh, *Numeric-analytic solutions of mixed-type systems of balance laws*, Applied Mathematics and Computation. vol. 265, pp. 133-143, 2015.
24. E. Az-Zo'bi, *On the reduced differential transform method and its application to the generalized Burgers-Huxley equation*, Applied Mathematical Sciences, vol.8, no. 177, pp. 8823-8831, 2014.
25. M. Banga, A. Bansal, and A. Singh, *Implementation of machine learning techniques in software reliability: A framework*, International Conference on Automation, Computational and Technology Management (ICACTM), pp. 241–245, 2019.
26. H. Pham, *System software reliability*, Springer Science Business Media, 2007.
27. M. Tashtoush, Y. Wardat, R. AlAli, K. Al-Saud, *The Impact of Cyberbullying on Student Motivation to Learn: In-Sights from Abu Dhabi Emirate Schools*, Humanities and Social Science Letters, vol. 11, no. 4, 461-474, 2023.
28. H. Pham, L. Nordmann, and Z. Zhang, *A general imperfect-software-debugging model with S-shaped fault-detection rate*, IEEE Trans. Reliab., vol. 48, no. 2, pp. 169–175, 1999.
29. H. Pham, *A new software reliability model with Vtub-shaped fault-detection rate and the uncertainty of operating environments*, Optimization, vol. 63, no. 10, pp. 1481–1490, 2014.
30. N. Anjum, M., Haque, M. A., and Ahmad, *Analysis and ranking of software reliability models based on weighted criteria value*, Int. J. Inf. Technol. Comput. Sci, vol. 19, no. 6, pp. 1–14, 2013.
31. S. H. Adel, K. S. Fatah, and M. S. Sulaiman, *Estimating the Rate of Occurrence of Extreme value process Using Classical and Intelligent Methods with Application: nonhomogeneous Poisson process with intelligent*, Iraqi J. Sci., pp. 3054–3065, 2023.
32. X. Yang and S. Deb, *Cuckoo search via Lévy flights*, 2009 World congress on nature and Biologically Inspired Computing (NABIC), pp. 210–214, 2009.
33. S. Yamada, M. Ohba, and S. Osaki, *S-shaped reliability growth modeling for software error detection*, IEEE Trans. Reliab., vol. 32, no. 5, pp. 475–484, 1983.
34. A. L. Goel and K. Okumoto, *Time-dependent error-detection rate model for software reliability and other performance measures*, IEEE Trans. Reliab., vol. 28, no. 3, pp. 206–211, 1979.
35. S. Yamada, M. Ohba, and S. Osaki, *S-shaped software reliability growth models and their applications*, IEEE Trans. Reliab., vol. 33, no. 4, pp. 289–292, 1984.
36. Y. Wardat, M. Tashtoush, R. Alali, A. Jarrah, *ChatGPT: A Revolutionary Tool for Teaching and Learning Mathematics*, EURASIA Journal of Mathematics, Science and Technology Education, vol. 19, no. 7, Article No: em2286, 2023.
37. S. Yamada, K. Tokuno, and S. Osaki, *Imperfect debugging models with fault introduction rate for software reliability assessment*, Int. J. Syst. Sci., vol. 23, no. 12, pp. 2241–2252, 1992.
38. H. Pham and X. Zhang, *An NHPP software reliability model and its comparison*, Int. J. Reliab. Qual. Saf. Eng., vol. 4, no. 3, pp. 269–282, 1997.
39. I. H. Chang, H. Pham, S. W. Lee, and K. Y. Song, *A testing-coverage software reliability model with the uncertainty of operating environments*, Int. J. Syst. Sci. Oper. Logist., vol. 1, no. 4, pp. 220–227, 2014.
40. B. Mohammad, I. Awan, H. Ugail, M. Younas, *Failure prediction using machine learning in a virtualised HPC system and application*, Cluster Computing, vol. 22, no. 2, pp. 471–485, 2019.
41. L. Li, *Software Reliability Growth Fault Correction Model Based on Machine Learning and Neural Network Algorithm*, Microprocessors and Microsystems, 80, 103538.