The Compatibility of Parametric Software Reliability Growth Models in PRGA

Reza Roshani[1], Homayun Motameni[2*], Hosein Mohamadi[3]

Abstract–Software Reliability (SR) is a key non-operational feature measured when evaluating software quality. To enhance this feature, it is important to detect failures and mitigate them in the testing phase. SR can be increased by identifying and removing this failure from the defect data. The existing literature consists of many models/methods applicable to measuring SR, including the SR Growth Models (SRGMs). Generally, SRGMs are in two main types: parametric and non-parametric. As these models are diverse, when applying to a certain problem, the particular requirements and conditions of that problem should be taken into account. The current paper explains the fundamental concepts of reliability, then reviews the Parametric SR Growth Models (PSRGMs) and evaluates various approaches already proposed in this domain. In addition, this study investigates the SRGMs compatibility by means of a novel Parallel Real-valued Genetic Algorithm (PRGA)-based method. The results achieved under a variety of conditions for each model showed the extent of compatibility with GA.

Keywords: SR Growth Model; Software Reliability; Genetic Algorithm; Compatibility.

1. Introduction

Regarding software quality, a non-functional requirement that should be taken into consideration is software reliability (SR). By definition, SR refers to the extent to which a computer program can execute a failure-free operation within a particular environment and during a specific time. Reliability function, $R(t)$, refers to the probability of non-failure during the time $(0, t]$,which could be represented as: $R(t) = P[N(t) = 0]$ [1].SR plays an important role when a software development team is making decisions about the development of software[2]. Such significance of SR has caused many scholars to propose different models for the estimation of this feature. The models developed in this regard could be classified into four groups: 1) time between failure models, 2) fault seeding models, 3) failure or fault count models, and 4) input-domain-based models [3, 4]. In this domain, 'failure' is defined as a software error or defect brought about by designer or programmer's mental error. A mental error can lead to a source code error. Any error in software can result in the crash of software under particular conditions in the course of the software operation[5]. Such situation can lead to a number of failures. Assume that S_i is the software failure's random variable at thei^{th} time$(i = 1,2,3, \dots)$. In such condition, $X_i = S_i - S_{i-1}$, $(i = 1,2,3, \dots)$; $S_0 = 0$ is a random variable that is the time interval between $(i-1)^{th}$ and i^{th} software failure. The conditional probability of non-occurrence of the i^{th} software failure in the interval $(t, t+x], (x \geq 0)$ provided that the $(i-1)^{th}$ software failure occurred at time t is as represented in Eq. (1):

$$R(X|t) = P\{X_i > x | S_{i-1} = t\} = \exp[-\{m(x+t) - m(t)\}], t \geq 0, x \geq 0 \qquad (1)$$

Software failure according to an input value and concerning specifications results in wrong output. Figure 1 depicts the way a software fault happened due to a particular input results in software failure [6].

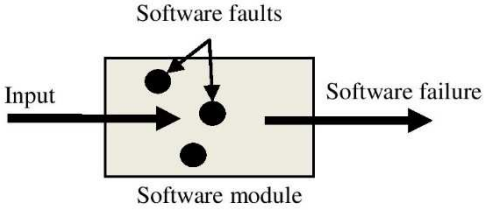

Fig. 1. Software fault and software failure

1 Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran. Email:r.roshany@gmail.com
2* CorrespondingAuthor: Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran.
Email: Homayun.Motameni@iau.ac.ir
3 Department of Computer Engineering, Azadshar Branch, Islamic Azad University, Azadshar, Iran. Email: h.mohamadi1983@gmail.com

Received: 2023.02.11; Accepted: 2023.03.09

SRGMs have been developed considering the failure occurred in the test phase [7]. In general, SRGMs fall into two groups: parametric and non-parametric. Non-Homogeneous Poisson Process (NHPP)-based parametric models are most extensively applied to the SR engineering field. Maximum Likelihood Estimation (MLE) and Least Square Estimation (LSE) are two parametric estimation models most widely used in the literature [8]. The homogeneous Poisson process models work on the basis of a Poisson process with rate$\lambda(t)$ with time-dependent [9]. SRGMs are applied to the measurement of the SR growth. The faults leading to failure could be well detected and repaired through determining the software failures. As a result, the software debugging processes can enhance SR. With the use of SRGMs, project managers would be capable of predicting more effectively the release time or required resources. For the prediction and estimation of SR, the desired model is required to be converted into a mathematical one. Two types of models have been proposed in the literature: failure rate and NHPP, which consider software as a black box in general and do not focus on the software components' internal structure and architecture[10]. The graph of the number of discovered faults based on the test time for these models could be concave or s-shaped, which represent two modes in Fig.2 [11].

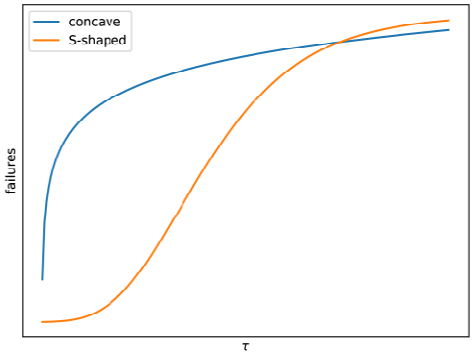

Fig. 2. Concave and S-shape model

The curves of the Concave and S-Shaped models show similar asymptotic behaviors with the increase in the number of detected failures; as failures are found and repaired in the course of the test phase, the number of such failure continuously reduces [12].

A number of SRGMs have been recently proposed in the relevant literature, which leads to higher levels of SR. Different models developed in this domain have some advantages and some drawbacks; as a result, for each problem in hand, the most appropriate model should be selected; otherwise, the release time would be prolonged and the project cost would rise[13]. There is not any single SRGM applicable to all problems and datasets[14, 15].

The present study assesses different SRGMs' performance in terms of SR growth estimation in order to determine the compatibility degree of each SRGM with GA. It employs the same parallel GA for the estimation of the parameters of all these models and evaluation of numerous criteria, e.g., accuracy and estimation time in various conditions.

The remaining of the paper is organized as follows: Section 2 presents the background of the study. Then, Section 3 elaborates on the proposed method of SRGM. Next, Section 4 analyzes the results obtained. To end with, Section 5 presents a discussion on findings, concludes the paper, and gives recommendations for future research in the relevant domain.

2. Background of the research

2.1 Reliability

Reliability refers to the probability of a system's success or the probability that the system can execute its defined task(s) under particular conditions. This non-functional feature also refers to the probability that a product (or part of a product) works properly for a specific time interval under certain conditions. In some cases, the concept of reliability is employed to measure the success of a system in terms of arranging for proper services. Reliability, $R(t)$, is referred to as a system's success probability in the time interval from 0 to t, as given in Eq. (2)[6]:

$$R(t) = P(T > t) \quad t \geq 0 \tag{2}$$

Where Tstands for a random variable to represent time to failure or failure time. On the other hand, unreliability,$F(t)$, measures the probability of failure at time$t$, using Eq. (3)

$$F(t) = P(T \leq t) \qquad for\ t \geq 0 \tag{3}$$

In fact, $F(t)$denotes failure's distribution function if the random variable of failure time$T$ consists of a density function$F(t)$; in such condition:

$$R(t) = \int_{t}^{\infty} f(s)\, ds \tag{4}$$

And its density function in terms of$T$given by Eq. (5):

$$\lim_{\Delta t \to 0} P(t < T \leq t + \Delta t) \quad (5)$$

This equation stands the probability of failure time between the operation time t and the subsequent operation time$t + \Delta t$. Assume that a new system exists, which has been tested successfully and is working well at time$t = 0$. Any increase in the time interval reduces the probability of system's success. Reliability is in fact a function of the time interval; initially, the system's operation is executed with the probability of 1, which eventually lowers down to the probability of0 .Thus, $R(\infty) = 0$, which means that the probability that a system succeeds in an infinite time interval is 0. For instance, in case of a system in the last 24 h, its reliability may equal 0.98. However, we cannot consider the system's reliability generally as 0.98 since the time interval is not defined.

2.2 Models for the estimation of software reliability

In the testing or execution phases of software, estimation methods are employed where there is failure information. In addition to testability, the software must have touched a level of maturity and had no need for any significant changes. The literature comprises numerous models developed for the SR prediction. Considering some common features, the models could be categorized mainly into four categories: 1) Time-between failure models: these models evaluate the reliability of a system through the estimation of the time between failures; 2) Failure or Fault count models: they operate based on the number of bugs existing in the time interval; 3) Fault seeding models: they predict the number of injection faults following the test operation through knowing the injection defects value; and 4) Input-domain-based models: these models attain random test cases in the validation phase from the operational profile inputs, then estimate the reliability of the system with the number of failures [4]. In this process, they divide the input space into a number of discrete classes, then assign a probability to each class depending on its chance for being chosen[16].LSE and MLE, described below, are among the most popular methods used widely by researchers for the estimation of reliability parameters.

LSE, which is known as a widely-used numerical method, estimates the parameters of a statistical model which minimizes the residual sum of squares between the expected data and the actual data. For instance, the residual sum of squares between the actual data and the expected data from a function $f(x)$with two parameters a and b is calculated using Eq. (6)[17]:

$$E(a,b) = \sum_{(i=1)}^{n} (y_i - f(x_i))^2 \quad (6)$$

When the differential value of $E(a,b)$, for instance, $\frac{d(E(a,b))}{da}$ and $\frac{d(E(a,b))}{db}$ is near zero, the estimated values a and $b$are considered the best solutions.

MLE, which is also an extensively-used numerical method, estimates the parameters of the statistical model which maximizes the value of the likelihood function. To implement MLE, there is a need for specifying a likelihood function from the statistical model. For instance, the probability function of a function$f(x)$with two parameters a and b is calculated using Eq. (7)[18, 19]:

$$L(a,b|x_1,\dots,x_n) = \prod_{i=1}^{n} f(x_i|a,b) \quad (7)$$

MLE determines the best numerical values of the model parameters that match the failure data. This method comprises two types of common failure data: failure time and the number of failures. The failure time, T, is represented as a vector of failure times, as given in Eq. (8):

$$T = < t_1, t_2, \dots, t_n > \quad (8)$$

where n stands for the number of observed failures. The number of failures is given in Eq. (9):

$$T = \{(t_1,k_1),(t_2,k_2),\dots,(t_n,k_n)\} \quad (9)$$

where t_i denotes the end of the i^{th} time interval, and$k_i$stands for the number of errors detected in thei^{th} time interval.

2.3 Test Effort Function (TEF)

Test Effort Function (TEF) is capable of describing the change in SR over time. SR generally much deals with the amount of effort for the identification and correction of software errors[20].TEFs have been offered in the literature in different types such as logistic, logistic generalization, and Weibull (see Table 1). TEFs are typically implemented based on consumption resources, e.g., human resources (person-hours), processor time, and number of test cases, in order to enhance the diagnosis efficiency of software.

In Table 1, $W(t)$stands for the cumulative effort test in the interval$(0,t]$, $f(t)$denotes the learning function, and $w(t)$ signifies the growth rate function of the test effort at

time t. Weibull suffers from a hasty growth problem in software development, which is not suitable to describing S-Shaped growth trends.$W(0)$in Logistic-TEF does not equal 0 at time $t = 0$, which is not true in real conditions. In addition, Chatterjee's drawback is its high time complexity[21].In the methods introduced in the current paper, the Weibull curve and exponential Weibull effort functions are used.

Table 1. Test effort functions

TEFs	W(t)
Exponential	$N(1-\exp(-\beta t))$
Generalized exponential	$N(1-\exp(-\beta t))^{\theta}$
Weibull curve	$N(1-\exp(-\beta t^{m}))$
Exponential Weibull	$N(1-\exp(-\beta t^{m}))^{\theta}$
Logistic	$\frac{N}{1+A\exp(-\beta t)}$
Generalization logistic	$\frac{N}{\sqrt[k]{1+A\exp(-kut)}}$
Log-logistic	$\frac{N(bt)^{c}}{1+(bt)^{c}}$
Chatterjee	$w(t)=W'(t)=\frac{k}{(f(t))^{P}}$

2.4 Software Reliability Growth Models

Software Reliability Growth Models (SRGMs) essentially estimate SR by the detection of defects in the testing phase. These extensively-used mathematical models are able to not only monitor, estimate, and evaluate SR, but also predict a software product's release time [22]. SRGMs comprise different parameters, e.g., the defect detection rate or the total number of defects present. In these models, the software's failure behavior is based on time; as a result, by moving into the future and executing tests and bugs, reliability could rise. This, in turn, could result in a decrease in the number of remaining failures or the time interval failures. The reason is that in SRGMs, the number of remaining errors reduces over time [11]. A statistical review of the literature shows that amongst all the models already proposed for the SR modeling purposes, SRGMs are the most-cited models. As the majority of the studies carried out in this domain have been devoted to themselves, this category of models is particular[10]. Eight types of SRGMs are employed in the present paper. Table 2 shows the names of the models and references. The parameters and symbols used in this table are defined as follows:

m(t),which denotes the failure detection phenomenon, is modeled with the mean value function. This shows the number of failures that are expected to occur in time $(0, t]$[1].

Type, which indicates the chart type of model, can be either S-Shape or Concave signified S and C, respectively.

For the mean value function, each model has a different combination of symbols and parameters, which are presented as follow:

a, total number of expected failures.

b, scaling parameter for failure detection rate.

c, a constant parameter of an exponential distribution.

$\emptyset$, test failure detection rate.

η, constant parameter in logistic learning function.

$X(t)$, cumulative test effort in the time interval $(0, t]$.

$\propto$, constant number parameter.

β, scale constant parameter.

p_i, failure ratio of generation i.

k, power of test effort function.

Table 2. Summary of SRGM used in this study

Model	Type	m(t)
Goel–Okumoto (GO) [23]	C	$a(1-e^{-bt})$
Yamada delayed S-Shaped (YDSS) [24]	S	$a(1-(1+bt)e^{-bt})$
Generalized Goel (GG) [25]	C	$a\left(1-e^{-bt^{c}}\right)$
Inflection S-shaped (INFS)[26]	S	$\frac{a(1-e^{-bt})}{1+\emptyset e^{-bt}}$
Flexible Non -Homogeneous Poisson Process (FNHPP) [27]	S, C	$ap\left(1-e^{-bX(t)}\right) + a(1-p)\frac{1-\left(1+bX(t)\right)e^{-bX(t)}}{1+\eta e^{-bX(t)}}$
Asraful-Nesar (AN) [7]	S	$\frac{a}{1+\beta e^{\frac{-bt}{\alpha}}}$
Pham–Zhang (PZ) [28]	S, C	$\frac{1}{1+\beta e^{-bt}}\left[(c+a)(1-e^{-bt}) - \frac{a}{b-\propto}(e^{-\propto t} - e^{-bt})\right]$
Li–Yi (LY) [29]	C	$a\left(1-p_1e^{-bt}-p_2e^{-bt-e^{-bt}+1} - (1-p_1-p_2)e^{-bt+e^{1-e^{-bt}}-1}\right), k = 0$ $\|\| a\left(1-e^{-bt^{k+\frac{1}{k+1}}}\right), p_1=1, p_2=p_3 = 0$

For instance, the GO model comprises two parameters of a and b[8]. the former denotes the total number of failures and the latter denotes the failure detection rate. The actual software defects and predicted defects of this model are presented in Fig.3[30]. Similarly, other concave models can estimate the SR.

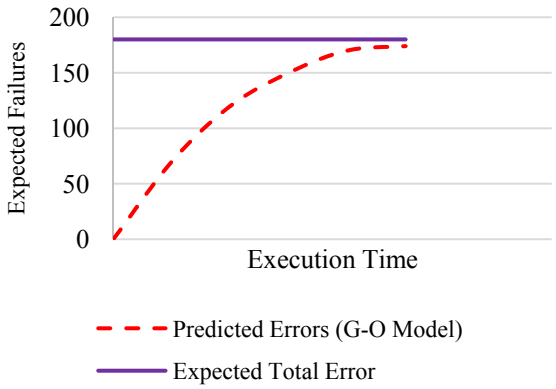

Fig. 3. Expected failure in GO model

3. The research method

The SRGMs' parameters could be estimated by different methods, among which is applying real-valued GAs[31]; therefore, their effectiveness in reliability growth models was examined in this study. To this end, a real-valued GA is applied to the earlier-mentioned SRGMs. The GA proposed for the solution of the problem using the genomes approach run in a number of stages until the stopping criteria are met. Each chromosome in GA denotes a candidate solution; the chromosomes are modified by three operators, i.e., selection, combination, and mutation in order to explore an appropriate solution to the problem. GAs normally comprise the following steps:

1- Setting the parameters, i.e., population size, maximum step, and the genetic operators rate.
2- Generating the initial candidate population in the shape of several chromosomes each of which stands for a solution[32].
3- Using the Fitness function to compute each chromosome's fitness.
4- Check the stopping criteria, which shows the termination of the algorithm operation else the next step continues.
5- Applying the three operators, i.e., selection, combination, and mutation.
6- Repeating the operation from Step 4 until the stopping criteria are met.

The real-value method is employed in order to display the chromosomes within the proposed algorithm. Table 3 presents the number of chromosomes in the population as the real-value for parameters a and b in the GO model.

Table 3. Representation of chromosome with real value

#	a	b
Chromosome0	1.27	4.21
Chromosome1	4.59	4.49
Chromosome2	3.79	4.70
Chromosome3	4.44	4.08
Chromosome4	0.34	0.01

The GA-based algorithm developed in this paper for the compatibility evaluation purposes works based on parallel implementation. The mentioned SRGMs' parameters are estimated by means of the same PRGA method. Figure 4 displays the overall process of the proposed method. As the figure depicts, initially the genetic parameters such as the number of processors, population size, and values of genetic operators are set. The $C_0, C_1, C_2, \dots, C_n$ stand for the chromosomes, and $< P_0, P_1, P_2, \dots, P_n >$ denote the parameters of each model considered as chromosome genes. As the proposed method is used in parallel, a topology type is needed to be selected for the interaction and cooperation of the Processing Element (PE)[33]. Based on the Hypercube 2D topology selected in this study, four processing elements are required to communicate form. Within the parallel model, the GA population comprises the sum of the sub-populations; as a result, the decentralized technique of initial population generation is employed in this study for the formation of a diverse sub-population[34]. Each processing element possesses its sub-population to which genetic operations are applied. Migration genetic operation is the related transfer of the superior chromosomes to another processing element[35]. In the present paper, *M* signifies the migration operation.

The environment implemented for the evaluation of these models are displayed in Fig. 5. Due to parallelization, the proposed method was designed and implemented using the Message Passing Interface (MPI) in the C# programming language and the Visual Studio 2020 environment. Within this environment, the desired data set, SRGM model type, and GA parameters could be determined. After these values are set properly, the algorithm begins estimating the parameters for the desired SRGM. At the final step, all the outputs achieved are stored in a file to be viewed and used later.

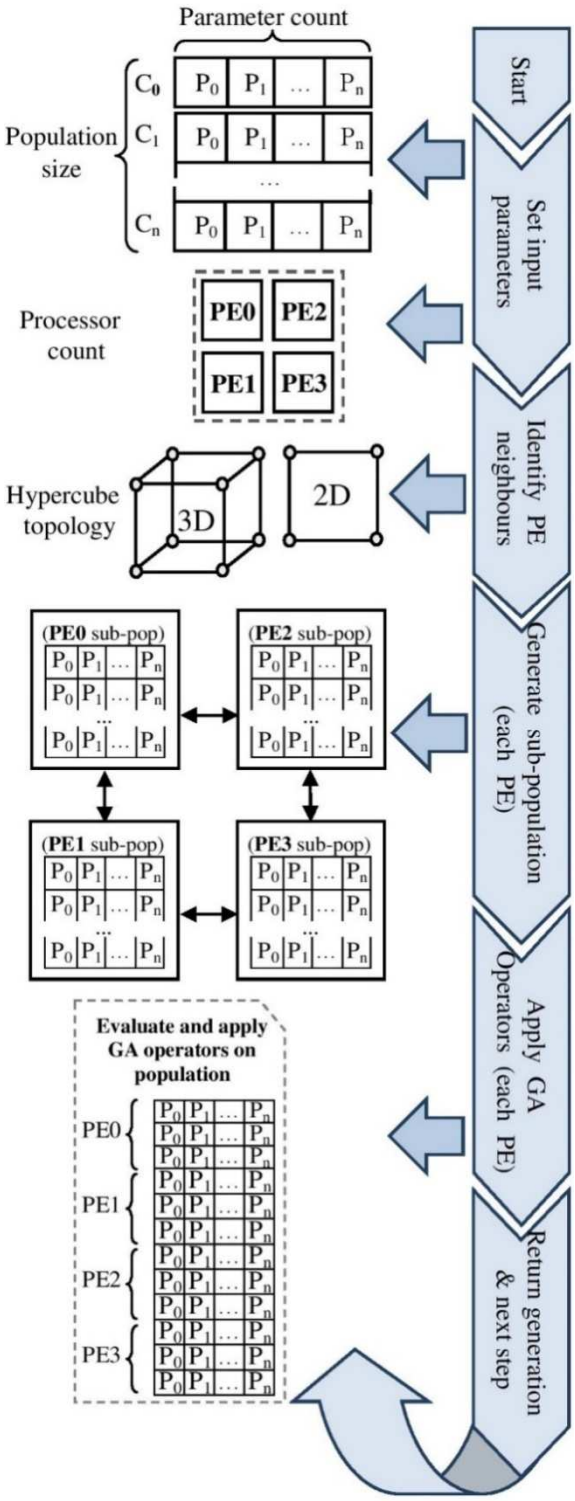

Fig. 4. Overview of the proposed algorithm

Figure 6 displays a sample of the output attained by running the proposed GA on the GO model in the DS1 dataset with the determined parameters. As the figure shows, the population size was fixed at 1000, the maximum number of execution steps at 1000, and Crossover, Mutation, and Migration rates equaled 25%, 2%, and 3%, respectively. The Weibull curve and Weibull exponential test function, as noted earlier, were used for the Mutation operator.

The proposed method was applied to estimating the parameters of all SRGMs and determining the extent to which each SRGM was compatible with the proposed GA-based algorithm. The outputs achieved are applied to the evaluation process in the following section.

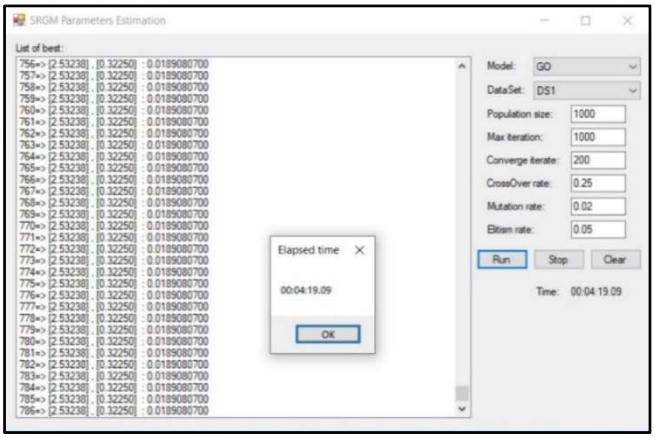

Fig. 5. The implemented environment

```
PE [0]:Iterate [0] => [2.20879], [11.35266]: 0.01791231
PE [2]:Iterate [0] => [2.28666], [0.99429]: 0.01818956
PE [3]:Iterate [0] => [2.28666], [0.99429]: 0.01818956
PE [1]:Iterate [0] => [2.61909], [0.57772]: 0.01822374
….
PE [1]:Iterate [999] => [2.38013], [0.58215]: 0.01857908
PE [3]:Iterate [999] => [2.38013], [0.58215]: 0.01857908
PE [0]:Iterate [999] => [2.38013], [0.58215]: 0.01857908
```

Dataset:	DS1
Model:	GO
Population size:	1000
Max iteration:	1000
Converge iterate:	200
Crossover rate:	0.25
Mutation rate:	0.02
Elitism rate:	0.05
Migrate rate:	0.03
Elapsed time:	00:04:19.09

Fig. 6. Sample of output the proposed method

4. Experimental result

This section explains the results obtained in this study. A model capable of making the best estimates with less error will have higher compatibility compared to GA. To measure the how compatible the GAs are with SRGMs, the 8 SRGM models presented in Table 2 are used in this paper. In addition, Table 4 presents the specifications of the data set employed; each data set is applied to various applications.

Table 4. List of datasets

Dataset	Application
DS1	On-line IBM entry software package
NTDS	U.S. Navy Fleet Computer Programming Center
J1	Project1 spacecraft system testing
SYS1	System failure1

For the evaluation of SRGMs, eight criteria were taken into account, which are explained as follow:

1) Elapsed Time (ET): It shows the average parallel execution time of processors in seconds, as given in Eq. (10):

$$ET = \frac{\sum_{i=0}^{P_n} EndOfTime_i - StartOfTime_i}{P_n} \quad (10)$$

where P_n stands for the number of processors, and i denotes the current processor.

2) Accuracy (ACC):this criterion shows the BestFitness value achieved among all the generations. It is calculated using Eq. (11):

$$ACC = \underset{1 \le i \le G}{\text{Max}}(BestFithness) \quad (11)$$

where $G$is the total number of executed generations of the algorithm, and the BestFitness is, in turn, computed with the following equation:

$$BestFitness = \frac{1}{\sum_{i=1}^{n}(m_i - m(t_i))^2} \quad (12)$$

where m_i signifies the number of actual defects observed at time t and $m(t_i)$ stands for the number of predicted defects at time$t$. ACC is taken into account as the chromosome with the best fitness value.

3) Mean Square Error (MSE)[36]:It indicates the mean errors between the actual defect data and the estimated defect data from SRGM. Equation (13) calculates this criterion:

$$MSE = \frac{\sum_{i=1}^{n}(m_i - m(t_i))^2}{n - p} \quad (13)$$

where n and p denote the size of the data set and the number of parameters used in the model, respectively. Note that lower values of MSE show the better performance of the model.

4) RootMean Square Error (RMSE): RMSE is the root of the MSE, which is applied to the computation of the difference between the actual observed values and the estimated ones. This criterion can effectively compare the prediction errors within a data set [37].It is calculated using Eq. (14):

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(m_i - m(t_i))^2} \quad (14)$$

5) Sum of Absolute Error (SAE):It refers to the sum of all the distances between the actual data and the predicted value[38].SAE can be computed using the following equation:

$$SAE = \sum_{i=1}^{n}|m_i - m(t_i)| \quad (15)$$

Mean Error Of Prediction (MEOP):This criterion is applied to the measurement of the standard deviation and also the calculation of the distance of the total error of SAE considering the value estimated by a model. Equation (16) can be used to calculate MEOP[38, 39]:

$$MEOP = \frac{\sum_{i=1}^{k}|m_i - m(t_i)|}{n - p + 1} \quad (16)$$

Remember that a model could be accurate and, at the same time, incorrect, and vice versa. As a result, Theil's Statistic (TS) is applied to correctness, while Prediction Relative Error (PRE) is applied to accuracy.

7) TS:It shows the mean percentage of deviation in all the data points. The closer the TS value to 0, the higher the correctness of estimation and model. TS is computed using the following equation:

$$TS = \sqrt{\frac{\sum_{i=1}^{k}(m(t_i) - m_i)^2}{\sum_{i=1}^{k}{m_i}^2}} * 100\% \quad (17)$$

8) PRE: It shows the ratio of the number of actual errors to the number of predicted errors at the failure prediction time, as given in Eq. (18):

$$PRE = \frac{Predicted - Actual\ Defects}{Predicted} \quad (18)$$

Note that a prediction will be acceptable if the TS value is less than 10% and the PRE value is in the range $[-10\%, +10\%]$ of the actual number of the defects [40].

Table 5 evaluates each model by the calculating criteria in the DS1 data set. The minimum total value indicates that

a model has the highest compatibility with GA. Findings show that AN has the lowest total value, followed by GO and then the other models. Table 6 gives the results of the specified criteria in the NTDS dataset. Within NTDS, the AN model again shows the lowest total and, consequently, is selected as the most compatible model with the parallel GA, which is followed by FNHPP in this regard.

Table 5.results of criteria and ranking in the DS1 dataset

SRGM	Criteria	Value	Criteria	Value	Sum
GO model	ET	3.22	SAE	24.3384	40.7390
	ACC	0.018887	MEOP	1.2169	
	MSE	2.5185	TS	7.8494	
	RMSE	1.5870	PRE	-0.0099	
YDSS model	ET	3.21	SAE	24.4309	41.1827
	ACC	0.018714	MEOP	1.2215	
	MSE	2.5445	TS	8.1892	
	RMSE	1.5951	PRE	-0.0273	
GG model	ET	3.94	SAE	24.3686	41.4873
	ACC	0.018892	MEOP	1.2184	
	MSE	2.5180	TS	7.8439	
	RMSE	1.5868	PRE	-0.0074	
INFS model	ET	3.46	SAE	24.2159	40.8335
	ACC	0.018912	MEOP	1.2108	
	MSE	2.5176	TS	7.8380	
	RMSE	1.5867	PRE	-0.0143	
FNHPP model	ET	6.94	SAE	22.5121	42.5261
	ACC	0.018892	MEOP	1.1256	
	MSE	2.5421	TS	7.7852	
	RMSE	1.5944	PRE	0.0078	
AN model	ET	4.73	SAE	23.9089	39.1706
	ACC	0.018775	MEOP	1.1954	
	MSE	2.4744	TS	5.2742	
	RMSE	1.5730	PRE	-0.0041	
PZ model	ET	7.31	SAE	24.4482	45.1769
	ACC	0.018775	MEOP	1.2224	
	MSE	2.5363	TS	8.0817	
	RMSE	1.5926	PRE	-0.0330	
LY model	ET	7.47	SAE	23.9512	45.6212
	ACC	0.018911	MEOP	1.1976	
	MSE	2.8953	TS	8.4124	
	RMSE	1.7016	PRE	-0.0257	

Figure 7 compares the mean execution times of SRGMs. This value was attained through running each model in all data sets. Generally, this value indicates which SRGMs have higher compatibility with the parallel GA regarding the time. As the figure clearly shows, the GO's elapsed time is less than that of the others.

To Check the overall consistency, these values were run for each model on all the defined data sets in a way to determine the models' general compatibility. In this regard, lower values show less difference and more compatibility with the parallel GA. As Figure 8 depicts, AN offers the minimum value, 72.03; therefore, this model is recognized to have the highest compatibility with the parallel GA, which is followed by FNHPP with 78.13.

Table 6.results of criteria and ranking in the NTDS dataset

SRGM	Criteria	Value	Criteria	Value	Sum
GO model	ET	4.05	SAE	69.4402	98.6999
	ACC	0.000014	MEOP	2.1043	
	MSE	7.8796	TS	0.2518	
	RMSE	2.8071	PRE	0.0169	
YDSS model	ET	4.31	SAE	68.3891	96.4733
	ACC	0.000016	MEOP	2.0724	
	MSE	6.0958	TS	0.1948	
	RMSE	2.4690	PRE	0.0121	
GG model	ET	5.75	SAE	73.1794	108.6913
	ACC	0.000013	MEOP	2.2176	
	MSE	7.3283	TS	0.2342	
	RMSE	2.7071	PRE	0.0247	
INFS model	ET	4.46	SAE	70.8755	101.0088
	ACC	0.000017	MEOP	2.1477	
	MSE	7.2594	TS	0.2320	
	RMSE	2.6943	PRE	-0.0402	
FNHPP model	ET	5.75	SAE	67.9421	86.9245
	ACC	0.000014	MEOP	2.0589	
	MSE	7.2153	TS	1.2846	
	RMSE	2.6861	PRE	-0.0125	
AN model	ET	4.96	SAE	64.2901	79.3622
	ACC	0.000017	MEOP	1.9482	
	MSE	5.5894	TS	0.1786	
	RMSE	2.3642	PRE	0.0317	
PZ model	ET	6.62	SAE	69.6103	91.6677
	ACC	0.000017	MEOP	2.1094	
	MSE	9.8613	TS	0.3152	
	RMSE	3.1403	PRE	0.0112	
LY model	ET	5.98	SAE	75.6352	96.1960
	ACC	0.000017	MEOP	2.2920	
	MSE	8.6225	TS	0.8151	
	RMSE	2.9364	PRE	-0.0852	

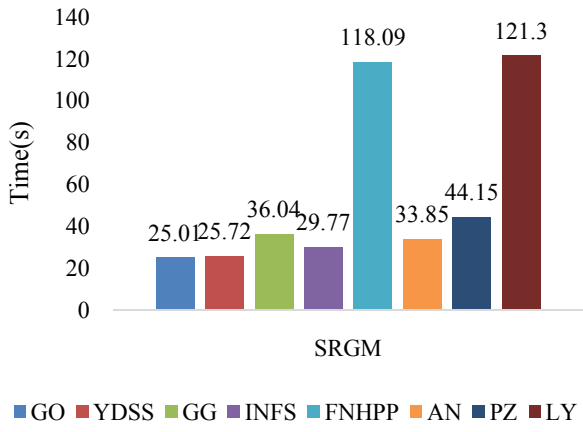

Fig. 7. The mean execution time of SRGMs

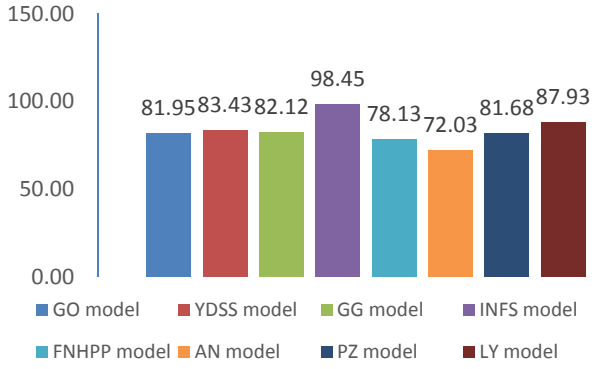

Fig. 8. The mean difference of SRGMs with PRGA

5. Conclusion

Reliability plays a critical role when examining the quality of software. Several models have been proposed in the literature to measure and estimate software reliability (SR), among which SRGMs is one of the most-used models. The current paper generally reviewed the relevant literature on the reliability growth and basic, novel models. The models examined in this study are among the most well-known ones in this domain. In addition, many methods have been developed for the estimation of these models' parameters, including GAs. The present paper was mainly aimed at investigating the extent each model is compatible with GA. Therefore, a parallel genetic method was applied to estimating the SRGM's parameters. This study took into consideration various criteria, e.g., run time, accuracy level, and MSE. To end with, each model's efficiency was evaluated through which the models with fewer differences and higher compatibility with GA were identified. The findings revealed that AN offered the highest level of compatibility with GA and had fewer errors, which was followed by FNHPP. As a result, AN and FNHPP are recommended to be applied to parameter estimation processes using GA.

References

[1] David D Hanagal and Nileema N Bhalerao, *Software Reliability Growth Models*, Springer, Singapore, 2021.

[2] Yeu-Shiang Huang, Kuei-Chen Chiu and Wan-Ming Chen, "A software reliability growth model for imperfect debugging," *Journal of Systems and Software*, Vol.188:111267(1)-111267(15), 2022.https://doi.org/10.1016/j.jss.2022.111267

[3] Ajeet Kumar Pandey and Neeraj Kumar Goyal, *Early software reliability prediction*, Springer, 2015.

[4] Shigeru Yamada, *Software reliability modeling: fundamentals and applications*, Vol.5, Springer, 2014.

[5] Mirmorsal Madani, Homayun Motameni and Reza Roshani, "Fake News Detection Using Feature Extraction, Natural Language Processing, Curriculum Learning, and Deep Learning," *International Journal of Information Technology & Decision Making,* 2023. https://doi.org/10.1142/S0219622023500347

[6] Hoang Pham, *System software reliability*. Springer Science & Business Media, Springer London, 2007.

[7] Md Asraful Haque and Nesar Ahmad, "An effective software reliability growth model," *Safety and Reliability*, Vol.40, No.4:209-220, 2021.https://doi.org/10.1080/09617353.2021.1921547

[8] M. Zubair Ahmad and N. Ahmad,"Parametric Software Reliability Growth Model with Testing Effort: A Review,"*in2021 International Conference on Computational Performance Evaluation (ComPE)*. Shillong, India, 2021. https://doi.org/10.1109/ComPE53109.2021.9752232

[9] Fabio Q. B. da Silva, André L. M. Santos, Sérgio Soares, A. César C. França, Cleviton V. F. Monteiro and Felipe Farias Maciel, "Six years of systematic literature reviews in software engineering: An updated tertiary study," *Information and Software Technology*, Vol.53, No.9:899-913, 2011. https:/doi.org/10.1016/j.infsof.2011.04.004

[10] Felipe Febrero, Coral Calero, and M Ángeles Moraga, "A Systematic Mapping Study of Software Reliability Modeling," *Information and Software Technology*, Vol.56, No.8:839-849, 2014.https:/doi.org/10.1016/j.infsof.2014.03.006

[11] Ajeet Kumar Pandey and N. Goyal,"Early Software Reliability Prediction - A Fuzzy Logic Approach," in *Studies in Fuzziness and Soft Computing*, 2013.

[12] E. Kashyap and A. Rana,"A Comparative Study of S-shape and Concave Software Reliability Growth Models,"*in2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 2015, pp.1452-1455. https://doi.org/10.1109/CICN.2015.280

[13] Kezhong Lu and Zongmin Ma, "A modified whale optimization algorithm for parameter estimation of software reliability growth models," *Journal of Algorithms & Computational Technology*, Vol.15:1-14, 2021. https://doi.org/10.1177/17483026211034442

[14] Neha Miglani, Poonam Rana, and M. Scholar, "Ranking of Software Reliability Growth Models using Greedy Approach,"*Global Journal of Business Management and Information Technology,* Vol.1, No.2:119-124, 2011.

[15] Ritu Bibyan and Sameer Anand, "Ranking of Multi-release Software Reliability Growth Model Using Weighted Distance-Based Approach,"*inOptimization Models in Software Reliability*, Springer International Publishing, Cham, 2022, pp.355-373. https://doi.org/10.1007/978-3-030-78919-0_16

[16] Cheng-Gang Bai, "Bayesian network based software reliability prediction with an operational profilem," *Journal of Systems and Software*, Vol.77, No.2:103-112, 2005. https:/doi.org/10.1016/j.jss.2004.11.034

[17] M. Geetha B, Santhosh Kumar Rand Cristin T. Daniya, "Least Square Estimation of Parameters for Linear Regression," *International Journal of Control and Automation*, Vol.13, No.02:447-452, 2020.

[18] In Jae Myung, "Tutorial on maximum likelihood estimation," *Journal of Mathematical Psychology,* Vol.47, No.1:90-100, 2003. https:/doi.org/10.1016/S0022-2496(02)00028-7

[19] C. Ojeda, W. Palma, S. Eyheramendy, and F. Elorrieta, "Extending time-series models for irregular observational gaps with a moving average structure for astronomical sequences," *RAS Techniques and Instruments*, Vol.2, No.1:33-44, 2023.https://doi.org/10.1093/rasti/rzac011

[20] SMK Quadri and N Ahmad, "Software reliability growth modeling with new modified weibull testing–effort and optimal release policy," *International Journal of Computer Applications*, Vol.6, No.12:1-10, 2010.

[21] Cong Jin and Shu-Wei Jin, "Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization," *Applied Soft Computing*, Vol.40:283-291, 2016. https:/doi.org/10.1016/j.asoc.2015.11.041

[22] Md A Haque and Nesar Ahmad, "Key issues in software reliability growth models," *Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science),* Vol.15, No.5:741-747, 2022.https://doi.org/10.2174/2666255813999201012182821

[23] Amrit L Goel and Kazu Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE transactions on Reliability*, Vol.28, No.3:206-211, 1979.https://doi.org/10.1109/TR.1979.5220566

[24] Shigeru Yamada, Mitsuru Ohba, and Shunji Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Transactions on reliability*, Vol.32, No.5:475-484, 1983.

[25] A. L. Goel, "Software Reliability Models: Assumptions, Limitations, and Applicability," *IEEE Transactions on Software Engineering*, Vol. SE-11, No.12:1411-1423, 1985. https://doi.org/10.1109/TSE.1985.232177

[26] Mitsuru Ohba, *Inflection S-shaped software reliability growth model, in Stochastic models in reliability theory,* Springer: Berlin, Heidelberg, 1984, pp.144-162. https://doi.org/10.1007/978-3-642-45587-2_10

[27] PK Kapur, DN Goswami, Amit Bardhan, and Ompal Singh, "Flexible software reliability growth model with testing effort dependent learning process," *Applied Mathematical Modelling*, Vol.32, No.7:1298-1307, 2008. https://doi.org/10.1016/j.apm.2007.04.002

[28] Hoang Pham and Xuemei Zhang, "An NHPP software reliability model and its comparison," *International Journal of Reliability, Quality and Safety Engineering*, Vol.4, No.03:269-282, 1997.

[29] Fan Li and Ze-Long Yi, "A new software reliability growth model: multigeneration faults and a power-law testing-effort function," *Mathematical Problems in Engineering*, Vol.2016:1-13, 2016. https://doi.org/10.1155/2016/9276093

[30] Satish Kumar and Harish, "Estimate the Software Reliability with Recursive Go-Model," *International Journal of Research in Computer Applications and Information Technology (IJRCAIT)*, Vol.3, No.6:40-49, 2015.

[31] Taehyoun Kim, Kwangkyu Lee, and Jongmoon Baik, "An effective approach to estimating the parameters of software reliability growth models using a real-valued genetic algorithm," *Journal of Systems and Software*, Vol.102:134-144, 2015.https://doi.org/10.1016/j.jss.2015.01.001

[32] Mirmorsal Madani, Homayun Motameni, and Hosein Mohamadi, "KNNGAN: an oversampling technique for textual imbalanced datasets," *The Journal of Supercomputing*, Vol.79, No.5:5291-5326, 2023. https://doi.org/10.1007/s11227-022-04851-3

[33] Subodh Kumar, *Introduction to parallel programming*, Cambridge University Press, 2022.

[34] Reza Roshani, Homayon Motameni, and Hosein Mohamadi, "A decentralized method for initial populations of genetic algorithms," *The Journal of Supercomputing*:1-20 , 2023. https://doi.org/10.1007/s11227-023-05066-w

[35] Reza Roshani and Mohammad Karim Sohrabi, "Parallel Genetic Algorithm for Shortest Path Routing Problem with Collaborative Neighbors," *Ciência e Natura*, Vol. 37, No.0:327-333, 2015.https://doi.org/10.5902/2179460X20790

[36] Madani Mirmorsal, "Comparing learning algorithms in neural network for diagnosing cardiovascular disease,"*CoRR*, Vol.abs/1611.01678, 2016. https://doi.org/10.48550/arXiv.1611.01678

[37] Bejjam Vasundhara Devi and R. Kanniga Devi, "Software reliability models based on machine learning techniques: A review,"*inAIP Conference Proceedings*, Vol.2463, No.1, 2022, pp.020038. https://doi.org/10.1063/5.0080442

[38] Da H. Lee, In H. Chang, and Hoang Pham "Software Reliability Growth Model with Dependent Failures and Uncertain Operating Environments,"*Applied Sciences*,Vol.12, No.23:12383(1)-12383(14), 2022. https://doi.org/10.3390/app122312383

[39] Mohd Anjum, Md. Asraful Haque, and Nesar Ahmad, "Analysis and Ranking of Software Reliability Models Based on Weighted Criteria Value," *International Journal of Information Technology and Computer Science,* Vol.5:1-14, 2013.

[40] N. Ullah, M. Morisio, and A. Vetro,"A Comparative Analysis of Software Reliability Growth Models using Defects Data of Closed and Open Source Software,"*in2012 35th Annual IEEE Software Engineering Workshop*, Heraklion, Greece, 2012, pp.187-192. https://doi.org/10.1109/SEW.2012.26