Heliyon 5 (2019) e02082

Contents lists available at ScienceDirect

Heliyon

www.elsevier.com/locate/heliyon

Review Article

# A novel approach of NPSO on dynamic weighted NHPP model for software reliability analysis with additional fault introduction parameter

Pooja Rani [a,b,*], G.S. Mahapatra [c]

[a] Department of Computer Science & Engineering, National Institute of Technology Puducherry, India
[b] University of Washington, Seattle, WA 98195, USA
[c] Department of Mathematics, National Institute of Technology Puducherry, Karaikal 609609, India

ARTICLE INFO

*Keywords:*
Computer science
Software reliability
Failure prediction
Artificial neural network
NHPP
Particle swarm optimization

ABSTRACT

This paper presents software fault detection, which is dependent upon the effectiveness of the testing and debugging team. A more skilled testing team can achieve higher rates of debugging success, and thereby removing a larger fraction of faults identified without introducing additional faults. A complex software is often subject to two or more stages of testing that exhibits distinct rates of fault discovery. This paper proposes a two-stage Enhanced neighborhood-based particle swarm optimization (NPSO) technique with the assimilation of the three conventional non homogeneous Poisson process (NHPP) based growth models of software reliability by introducing an additional fault introduction parameter. The proposed neuro and swarm recurrent neural network model is compared with similar models, to demonstrate that in some cases the additional fault introduction parameter is appropriate. Both the theoretical and predictive measures of goodness of fit are used for demonstration using data sets through NPSO.

## 1. Introduction

The ever-increasing demand of computer software plays a crucial role not only in scientific and industrial enterprise, but also in our daily life. The development of reliable software products with increasing size and complexity is a very strenuous and challenging task. We require a software that must be reliable ([1, 2, 3]). The IEEE [4] defines software reliability as: "*The probability that software will not cause a system failure for a specified time under specified conditions*". Reliability is certainly the prime factor to be claimed for study of any discipline of engineering as it quantitatively measures quality, and the quantity can be properly engineered ([5, 6, 7]). Software reliability is a propensity to improve and it can be treated as a growth factor during the testing process ([4, 5, 7, 9, 10, 12, 13, 14]).

There is no universal model for software reliability prediction, rather every model has its own special functionality for better reliability prediction. The NHPP S-shaped model is shown to be very useful in fitting software failure data. The testing process of software reliability model considers fault detection ([8], [15], [16]) and fault isolation. In general, NHPP growth model with imperfect debugging ([7], [17], [18]) is one of the best kind of analytical framework, which is used to approximate mean value function (MVF) of exponential growth, considering that the failure occurrence is up to a certain point of time. During the past three decades, the NHPP growth curve model has been applied in different researches such as: evaluation of agriculture, exchange of transportation systems, development of discoveries, transformation of aircraft industry, the evolution of telecommunication systems and statistical modeling [19]. Therefore, it is difficult to find an area of system evolution, where NHPP growth curves analysis has yet not been applied upon. The NHPP growth modeling is very useful for interpretation with the progress of testing and debugging [13]. Several researchers have investigated NHPP based SRGMs in software reliability engineering ([5], [11], [10], [20], [21]). A pioneer attempt was made by Goel and Okumoto [22] in developing NHPP exponential curve for presenting the software failure phenomenon ([58], [59]). Jinyong [18] presented a nonlinear, NHPP imperfect software debugging model with fault introduction. The Artificial Neural Network (ANN) model is as an optimization tool to solve non-linear continuous function with arbitrary accuracy. Karunanithi et al. [23] presented that neural network models have more accurate endpoint prediction compared to some analytic models. Khoshgoftaar and Szabo [24] used ANN for prediction of number of faults occur at testing phase. Tian and Noore [25] explored an evolutionary ANN for software cumulative failure time prediction based on multiple-delayed-input and single-output architecture. Zheng [26]

* Corresponding author at: University of Washington, Seattle, WA 98195, USA.
*E-mail address:* prani@uw.edu (P. Rani).

https://doi.org/10.1016/j.heliyon.2019.e02082
Received 13 August 2018; Received in revised form 9 January 2019; Accepted 9 July 2019

described an ensemble of ANNs for prediction of software reliability. Su and Huang [27] proposed dynamic weighted combinational model (DWCM) based on ANN [60], [61] for prediction of software reliability. Kapur et al. ([28], [29]) presented generalized dynamic integrated model using ANN to incorporate learning phenomenon with complexity of faults. Qiuying [30] proposed a testing-coverage software reliability model that not only considers the imperfect debugging, but also the uncertainty of operating environments. Chuang et al. [31] presented dynamically weighted ensemble-based prediction system for adoptively modeling. Discrete SRGMs are needed for modeling discrete software failure data for better goodness of fit performance. Pooja and Mahapatra [32] proposed DWCM approach using ANN 2-stage architecture by combining the imperfect debugging models [62].

Many attempts have been introduced in literature to address the Neural Network (NN) ([33], [34], [35], [36], [37]) and PSO ([38, 39, 40, 41, 42, 43]) which offer promising approaches to software reliability estimation and prediction in NHPP modeling ([44], [21]). In this paper, in order to improve the ability of neural network to get away from confined on local optimum, the proposed PSO algorithm is applied to update the network based on neighborhood PSO.

PSO is a simple and efficient algorithm to solve non-linear optimization problem as it controls fewer parameters, and has better convergence performance compared to other soft computing paradigms [45]. Further in this field, Riccardo and James [39] defined additional variation in the algorithm. Bai [46] reviewed advantages and the shortcomings of PSO techniques. Eberhart and Shi [47] proposed the best approach of use of the constriction factor under maximum velocity constraint on each dimension. Li et al. [12] proposed a hierarchical mixture of software reliability models using an expectation–maximizing algorithm to estimate the prediction parameters. An adaptive inertia weight investigated by Xu [37]. Nasir et al. [48] presented dynamic neighborhood learning PSO. Wang et al. [40] proposed good balance between global search and local search by suggesting a trade-off between exploration and exploitation abilities. Sun et al. [41] presented constrained optimization problems using PSO algorithm. Mousa et al. [49] solved multi-objective optimization problem in local search based on hybrid PSO algorithm. Cong [50] explored quantum PSO algorithm, to optimize parameter of NHPP based SRGM testing effort function. Viet-Hung [51] introduced a robust optimization method based on differential evolution algorithm [52]. Huang [53] described particle-base simplified swarm optimization algorithm while considering updating mechanism to increase software reliability. Wang [54] proposed optimized method to improve NHPP model. In this paper, we propose a two-stage Enhanced NPSO technique based on the assimilation of three well known NHPP based software reliability growth models for software reliability prediction.

Imperfect debugging models can characterize the quality of software fault removal in addition to the rate of faults discovered. This paper approaches recurrent NN architecture on NHPP based software reliability growth model (SRGM), incorporating imperfect debugging phenomenon. We have modified fault introduction rate by taking into consideration the efficiency of testing team skill during software development. The paper focuses to identify the superiority of the use of the existing model proficiently, rather than developing a generalized and potentially complicated model. Consequently, a DWCM is proposed, which considers the large learning capability factor during the testing and debugging process. The novelty of this proposed work is to give more attention on fault introduction rate in imperfect debugging.

The rest of the paper is rolled out in the following way: Section 2 describes the proposed model derivation, and presents the applied methodology in detail. Section 2.2 introduces the proposed PSO with implementation of modified algorithm using neural networks. In Section 2.5, we define model validation and failure data description, along with the fitness function and predictive validity criteria. Section 2.8 gives the experimental evaluation carried out through the experiments, result, and summary. Finally, Section 3 concludes the paper based up on the outcomes of the analysis and prediction.

**Table 1**
NHPP imperfect debugging models with parameters and mean value function.

| Name of model | **a(t)** | **b(t)** | MVF **y(t)** |
|---|---|---|---|
| Yamada Model (Y) | $ae^{-\alpha t}$ | $b$ | $\frac{ab}{b+\alpha}(e^{\alpha t}-e^{-bt})$ |
| PNZ Model (P) | $a(1+\alpha t)$ | $\frac{b}{1+\beta e^{-bt}}$ | $\frac{a}{1+\beta e^{-bt}}\{\left(1-\frac{\alpha}{b}\right)\left(1-e^{-bt}\right)+\alpha at\}$ |
| Roy Model (R) | $a(\alpha-e^{-\beta t})$ | $b$ | $a\alpha(1-e^{-bt})-\frac{ab}{b-\beta}(e^{-\beta t}-e^{-bt})$ |

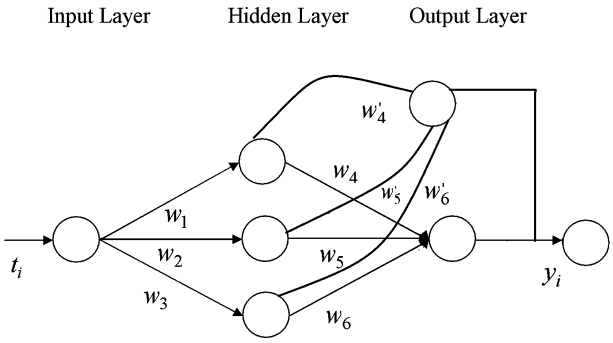


**Fig. 1.** PNSRNN architecture.

## 2. Main text

Due to lack of knowledge regarding the very particular debugging process for the specific software, the introduction rate of fault is higher and the count of errors increase at the starting of the testing stage. Based upon this practical idea, we have chosen software reliability models based on the fault introduction rate $\alpha$. The proposed model can be applied on the software reliability model having higher fault introduction rate at the starting compared to the end of testing phase. Error detection rate, error content, and the MVFs of NHPP based imperfect debugging software reliability models: the Yamada Model [6], Pham-Nordmann-Zhang (PNZ) model [17], and Roy Model [55], are presented in Table 1, where $a(t)$ denotes the total number of faults in the software, $b(t)$ denotes faults per unit of time, $\beta$ is the inflection factor, and $y(t)$ denotes MVF i.e. the expected number of faults detected by time $t$.

We consider Yamada model [6], where $\alpha$ is time-dependent exponentially increasing function, and nature of fault introduction rate is exponential. Further, we consider PNZ model [9] where $\alpha(t)$ is time-dependent, and nature of fault introduction rate is linearly increasing function, and also the Roy model [55] which presented that the fault introduction rate is constant.

### 2.1. Formulation of Neuro and Swarm Recurrent Neural Network model

The proposed Neuro and Swarm Recurrent Neural Network (PNSRNN) is constructed by combining the selected SRGMs to develop the DWCM as shown in Fig. 1.

The weights $(p, q, r, w_i > 0)$ represent the strength of inputs to hidden, and hidden to output layer. Here $w'_4$, $w'_5$ and $w'_6$ represent the proportion of expected number of latent faults; and $w_1$, $w_2, w_3$ denote the fault rate depending on time. We consider the MVF of NHPP imperfect debugging reliability models: Yamada Model $[a_1\frac{p}{p+q}[e^{qt_i}-e^{-w_1 t}]]$, PNZ Model $[a_2(\frac{(1-e^{-w_2 t_i})}{1+re^{-w_2 t_i)}}(1-\frac{q}{p})+qt_i)]$, and Roy Model $[a_3 q(1-e^{-w_3 t_i})-\frac{p}{p-r}(e^{rt_i}-e^{-w_3 t_i})]$. We can say that $q=\alpha$, $r=\beta$, $p=b$, $b_1=w_1$, $b_2=w_2$, $w_3=b_3$, $a_1=w'_4$, $a_2=w'_5$, $a_3=w'_6$. Here activation functions of proposed PNSRNN for three neuron in the hidden layer $\frac{p}{p+q}[e^{qt_i}-e^{-w_1 t}]$, $\frac{1}{1+re^{-w_2 t_i)}}\{1-e^{-w_2 t_i)}\}(1-\frac{q}{p})+qt_i$ and $q(1-e^{-w_3 t_i})-\frac{p}{p-r}(e^{rt_i}-e^{-w_3 t_i})$.

We need to predict $t_i$ if any constant is introduced in place of 1. We construct the proposed PNSRNN with back-propagation path to the hidden layer from the output layer. We finally get

$$\hat{y}(t_i) = w_4[\frac{p}{p+q}(e^{qt_{i-1}} - e^{-w_{1-1}t})] + w_5[\frac{(1-e^{-w_2 t_{i-1}})}{1+re^{-w_2 t_{i-1}}}(1-\frac{q}{p}) + qt_{i-1}] \quad (1)$$
$$+ w_6[q(1-e^{-w_3 t_{i-1}}) - \frac{p}{p-r}(e^{rt_{i-1}} - e^{-w_3 t_{i-1}})]$$
$$+ (w'_4 + w'_5 + w'_6)\hat{y}(t_{i-1})$$

The primitive function of output is $w_i$. The constants $w'_4\phi'(t)$, $w'_5\phi'(t)$, $w'_6\phi'(t)$ are divided into three units associated to the recurrent output unit, by introducing a 1 into the output unit, multiplying it by the stored value $\phi'(t)$, and distributing the result through the edges with weights $w_i$ to $m$ units. Therefore the network occurs backwards as the recurrent architecture demands, i.e. the algorithm works with networks of $m+1$ nodes [32]. According to equation (1) we have successfully derived the DWCM into proposed PNSRNN model.

To add, the randomly optimal algorithm is a set of techniques used to increase the success and speed. NPSO is a scholastic optimization method which is successfully applied to train RNN. In this study, conventional NN is used to modify the network parameter and precision ([56], [23]) to improve the ability of the network.

### 2.2. Particle swarm optimization

PSO comprises of a set of particles moving within the search space, and each particle represents a potential solution vector (fitness) of the optimization problem [57]. The velocity and position of each particle $i$ of $n$ dimensional is represented by vectors $V_i = (v_{i1}, v_{i2}, ..., v_{in})$ and $X_i = (x_{i1}, x_{i2,}..., x_{in,})$, respectively. A flock of agents optimize a particular objective function ([26], [46]). During search process, each individual knows its best value $Pbest_i$ and its position. Each individual knows the best value in the group $Gbest$ among $Pbest_i$. Let $Pbest_i = (pb_{i1}, pb_{i2,}..., pb_{in,})$ and $Gbest_i = (g_{i1}, g_{i2,}..., g_{in,})$ be the position of the $i$-th individual and its neighborhood's best position. Therefore, the modified velocity and the distance from $Pbest_i$ and $Gbest$ of each individual is as follows

$$V_i(t+1) = w.V_i(t) + c_1 r_1\left(Gbest(t) - X_i(t)\right) + c_2 r_2\left(Pbest_i(t) - X_i(t)\right) \quad (2)$$

where $V_i(t)$ and $V_i(t+1)$ is the current velocity and modified velocity of individual $i$ at iteration $t$ and $t+1$ respectively. $X_i(t)$ the current position of individual $i$ at iteration $t$; $w$ $(0 < w < 1)$ is inertia weight to equipoise exploration and exploitation in local and global search capabilities. $c_1$ and $c_2$ represent the cognitive and social learning factors; $r_1$ and $r_2$ are uniform random numbers generated in $[0, 1]$. $Pbest_i(t)$ and $Gbest(t)$ is the best position of $i$-th individual and group till iteration $t$ respectively. From the current position, each individual moves to the next one through the modified velocity obtained from equation (2) using the following equation:

$$X_i(t+1) = Pbest_i(t) + V_i(t+1) \quad (3)$$

### 2.3. Concept of neighborhood PSO

Conventional PSO suffers from premature convergence like other stochastic search algorithms while solving high multi-modal problems. The sub-optimal could be near to the global optimum, and the neighborhood of the trapped particle may contain global optimum. Such problems can be addressed by introducing the concept of neighborhood to find better optimal solutions [52].

The most common one is ring topology for implementation of the neighborhood, where each particle is assumed to be neighbors and is determined by radius of the neighborhood to 5. Let $P_i$, $i = 1, 2, ..., N$ be the $i$-th particle of the swarm for the swarm size N. The N particles are organized on a ring based on their index, such that 2 and $N$ are

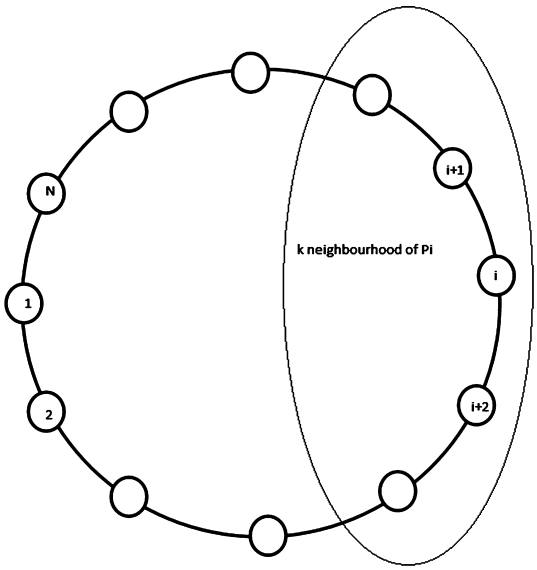


**Fig. 2.** Ring topology of PSO.

two immediate neighbors of particle 1. The k-neighborhood radius of the particle $(P)$, $i$ consists of particles $P_{i-k}, ..., P_i, ..., P_{i+k}$. Fig. 2 presents a ring topology with k-neighborhood radius. The neighborhood-based PSO algorithm, which considers global best particle in neighborhood of a particle in place of global best particle in velocity updating rule to avoid trapping in local optima.

Here, the first two adjacent particles on both sides of a particle in the ring are considered as neighbors. If the position of the best particle is $lbest$ in the neighborhood of the particle $P_i$, then equations (2) and (3) become as follows:

$$V_i(t+1) = wV_i(t) + c_1 r_1\left(pbest_i(t) - X_i(t)\right) + c_2 r_2\left(lbest(t) - X_i(t)\right) \quad (4)$$

$$X_i(t+1) = Pbest_i(t) + V_i(t+1) \quad (5)$$

Fig. 3 depicts the concept of standard PSO. It represents how each of the influence the components motion, memory and intelligence influence combine to result in an iterated particle velocity and subsequently the position. In the modified pPSO, influence the components inertia weight as motion $[wV_i(t)]$, memory $[c_1 r_1\left(pbest_i(t) - X_i(t)\right)]$ and intelligence $[c_2 r_2\left(lbest(t) - X_i(t)\right)]$ influence combine to result in an iterated particle velocity and subsequently the position. According to (5), the particle will shift towards its own best position, and also the best position of its neighborhood, instead of the global best position as in equation (4).

### 2.4. Fitness function

The proposed model is trained by proposed neighborhood based PSO to find out the optimal solution ([56], [32]). The weight and parameters of the PNSRNN are considered by $[w_1, w_2, w_3, w'_4, w'_5, w'_6, p, q, r]$. Here, the fitness function $N_{rmse}$ is described for the purpose, which is given by Normalized Root Mean Square Error (NRMSE) measurement for minimization of the error generated by the RNN as follows:

$$N_{rmse} = \sqrt{\frac{\sum_{i=1}^{n}\left(y'_i - y_i\right)^2}{\sum_{i=1}^{n} y_i^2}} \quad (6)$$

where $n$ denotes data points are applied to train the NN; $y_i$ and $y'_i$ represent the actual and predicted value for the $i$-th software failure data point. NRMSE value is minimized by the proposed PSO algorithm during learning of the RNN [56].

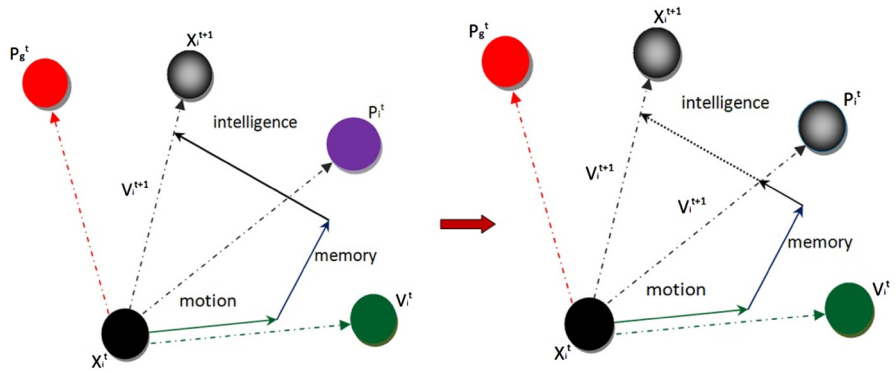


**Fig. 3.** Concept for searching standard PSO and pPSO.

Pseudo Code for training of PNSRNN based NPSO:

INPUT: Set parameters $X_i$ swarm size N, and $I_{max}$,

OUTPUT: Print optimum values of weights and parameters of proposed Model

S1. For $V_i = 1$ to N.

S2. Obtained $Y(t_i)$, $i = 1$ to N

S3. Calculate gbest randomly within the search boundaries.

S4. For each particle i = 1;2 to N do:

S5. Calculate the velocity of the particle vi.

S6. Determine the particle best known position as its initial position i.e., pi = xi.

S7. Calculate the particle global best among pi

S8. While ($t \leq I_{max}$), do:

S9. For each particle i = 1;2;:::;N do:

S10. Update $V_i < V_{i(min)}$ then $V_i = V_{i(min)}$;

S11. Update $V_i > V_{i(max)}$ then $V_i = V_{i(max)}$;

S12. Determine the local best particles position lbest in the neighborhood of the particle.

S13. Obtained the values of the parameters w; c1 and c2, and random numbers r1 and r2 in $[0, 1]$.

S14. Update the particles velocity vi and position xi from equation (4)

S15. Update the position xi from equation (5)

S16. Obtain $Y(t_i)$ equation (1).

S17. Update local best and global best.

S18. t is increment by 1.

S19. End For.

S20. End For

S21. End While

### 2.5. Failure data and model validation description

For checking the validity, the proposed model has been tested on three test data of software development projects. Software failure data is obtained in pairs $(t_i, y_i)$, where $t_i$ and $y_i$ denote the cumulative software execution time and corresponding cumulative number of failure respectively. Each data set is normalized before feeding to the ANN in the range $[0, 1]$.

**Data set-1 (DS1):** DS1 (Ohba [32]) was collected from real time command and control system during 21 days of testing and 46 faults were recorded.

**Data Set-2 (DS2):** DS2 (Wood [5]) was collected from Release 1 of Tandem Computer Project, in which 100 faults were detected for 20 weeks of duration.

**Data Set-3 (DS3):** DS3 (Telecommunication System Data [9]) was pair of the time of observation and the cumulative number of faults detected during 21 days of testing, measured on basis of the number of shifts spent running test cases, and analyzing the results 43 faults were detected.

### 2.6. Predictive validity criterion

The Relative Prediction Error (RPE) is the prediction ability of the future failure behavior from past and present failure behavior ([5], [32]). It is defined as follows:

$$RPE = \frac{\hat{m}(t_q) - m_q}{m_q} \tag{7}$$

Assuming the observed $m_q$ failures by the end of testing time $t_q$. The estimated values of parameters in the MVF provides the estimate of the number of failures $\hat{m}(t_q)$ by $t_q$. If the RPE value is positive and negative, then the SRGM is said to overestimate or underestimate the fault removal process respectively. A value close to zero indicates more accurate prediction, thus more confidence in the PNSRNN, however it is acceptable for within ±10%. Musa [2] presented that the software reliability models having the best predictive validity yielded the best values of other reliability quantities for that failure data.

### 2.7. End point prediction

The end-point prediction is to predict the number of failures which have been observed till the testing time $t_x$, and using the available failure data up to time $t_e (t_e \leq t_x)$. We apply different sizes of training patterns to train the ANN, and try to predict the number of failures at the end of the testing time $t_x$, and hence predictive validity can be checked by $RPE$ for different values of $t_e$ [2].

### 2.8. Experimental evaluation for performance analysis

In this section, the performance of proposed model is compared with NN and PSO based DWCM with the Yamada Model [6], PNZ Model [7] and Roy Model [56] in software reliability modeling and analysis. We have exhibited the performance of the models using Goodness of fit, convergence by fitness function, End point prediction, and RPE metrics.

### 2.9. Performance measures for DS-1

The fitting of the proposed model to data sets DS1 present in Fig. 4, which depicts the plots of the actual fault compared with predicted faults and the data estimated by the MVF of the proposed model.

From Fig. 4, we find the proposed model has magnificent prediction capability compared to observed values for Data Set 1.

Fig. 5 depicts the link of maximum number of iteration and excellent fitness outcome for proposed PNSRNN model. It is the training of the RNN through the test data obtained from software failure data set 1.

Table 2 shows that the proposed model has much better end-point prediction ability compare to NN, PSO and DWCM. PSO approach on PNSRNN has the most end-point predictive power at the end point result $E_8$. PNSRNN has the overall commanding ability for end-point predic-

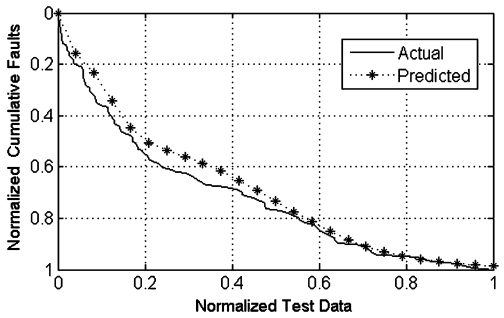


**Fig. 4.** Goodness of fit for DS1.

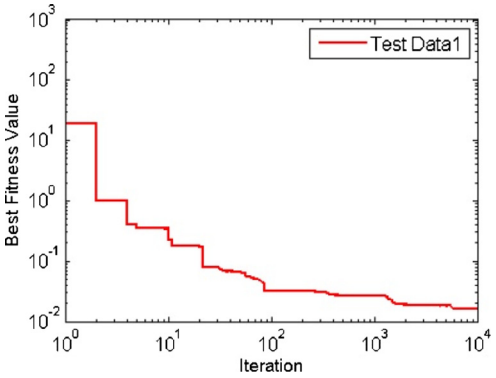


**Fig. 5.** Convergence of fitness values for DS 1.

tion than PSO and NN. The end-point predictive competence of the proposed model increases with execution time. The model is developed by combining two of three models has a better end-point predictive power than the models that are developed by NN, PSO and DWCM. It is obvious that for all end point results, the NPSO approach has a better end-point prediction capability than $NN_{YP}$, $PSO_{YP}$, $NN_{YR}$, $PSO_{YR}$, $NN_{PR}$, $PSO_{PR}$, NN, PSO and DWCM.

Fig. 6 depicts that the proposed model has smaller RPEs than the DWCM, NN, PSO and other existing models. The PNSRNN has much lesser RPE which establish excellent software reliability.

### 2.10. Performance measures for DS-2

The fitting of the proposed model to DS2 is presented in Fig. 7, which shows the plots of the actual fault compared with the predicted faults by the MVF of the proposed model.

Fig. 8 most effectively represents convergence graph for the NPSO during minimization of error generated from equation (6). It is training by failure data by DS-2 for PNSRNN model.

The comparison of end point result of proposed PNSRNN model is presented in Table 3. The end point value of the proposed model improves from the iterations $E_1$ to $E_{10}$, and reached lowest end point

**Table 2**
Comparison of end point result of PNSRNN model for DS 1.

| | $NN_{YP}$ | $PSO_{YP}$ | $NN_{YR}$ | $PSO_{YR}$ | $NN_{PR}$ | $PSO_{PR}$ | $NN$ | $PSO$ | $DWCM$ | $NPSO$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $E_{10}$ | 64.6795 | 49.1865 | 40.6612 | 37.6588 | 23.5535 | 31.6499 | 6.0170 | 0.6659 | 1.3082 | **0.0974** |
| $E_9$ | 48.7511 | 18.6534 | 26.9104 | 52.3012 | 6.5612 | 41.0519 | 3.4043 | 2.2461 | 1.1634 | **0.2679** |
| $E_8$ | 29.5798 | 28.3503 | 24.5119 | 47.8788 | 10.8851 | 21.5055 | 2.3177 | 5.2668 | 0.7847 | **0.2780** |
| $E_7$ | 44.1151 | 11.9057 | 17.0376 | 13.7094 | 15.6594 | 37.6405 | 3.2760 | 11.1002 | 2.9823 | **0.3293** |
| $E_6$ | 24.3763 | 20.2017 | 66.1641 | 17.0710 | 10.2457 | 17.5801 | 9.8850 | 8.6444 | 0.2759 | **0.3301** |
| $E_5$ | 31.5467 | 20.6991 | 24.2581 | 44.1102 | 34.8596 | 10.7967 | 13.9846 | 8.7973 | 1.7892 | **0.4233** |
| $E_4$ | 15.8331 | 27.4585 | 32.8564 | 15.3095 | 49.7422 | 28.1310 | 5.3551 | 9.9872 | 2.1241 | **0.4526** |
| $E_3$ | 30.6549 | 48.1234 | 19.9689 | 17.2630 | 18.4782 | 17.1943 | 6.8211 | **0.0230** | 3.5768 | 0.5503 |
| $E_2$ | 10.4981 | 17.9740 | 21.4475 | 17.1506 | 32.2898 | 26.3240 | 4.5263 | 2.9070 | 3.7622 | **0.6017** |
| $E_1$ | 18.3798 | 34.9055 | 18.8066 | 19.2588 | 30.8308 | 25.2519 | 9.2694 | 2.8378 | 6.8211 | **0.7035** |

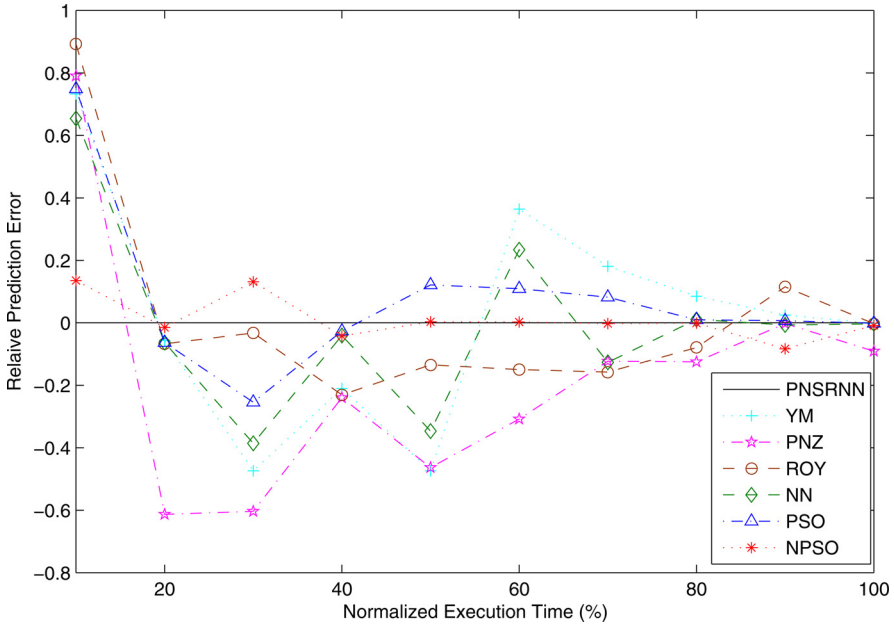


**Fig. 6.** RPE curve of PNSRNN model for DS 1.

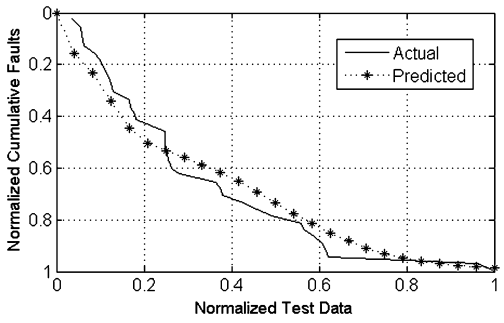


**Fig. 7.** Graph of goodness of fit for DS 2.

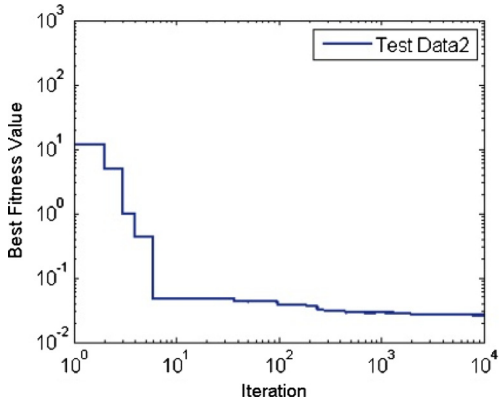


**Fig. 8.** Convergence of fitness values for DS 2.

value at $E_{10}$. The proposed model has better end-point prediction in $E_1$ to $E_8$. NPSO than NN, PSO and DWCM approach. NN approach on PNSRNN has the utmost end-point predictive ability at the End Point result $E_{10}$. The end point predictive powers are increased with the growth of execution time. The proposed model has best end point prediction by NPSO than NN, PSO an DWCM approach. NN approach has relative errors which are smallest from RE values of the model under comparison at end point result $E_{10}$ and $E_9$. It is obvious that for all end point results, the NPSO approach has better end-point prediction capability than $NN_{YP}$, $PSO_{YP}$, $NN_{YR}$, $PSO_{YR}$, $NN_{PR}$, $PSO_{PR}$, NN, PSO and DWCM.

Fig. 9 shows that the proposed model has smaller RPEs than the other than NN, PSO and DWCM. The PNSRNN has the lowest RPE compare to other models, i.e. the proposed PNSRNN has the best prediction capability than NN and PSO.

### 2.11. Performance measures for DS-3

The fitting of the proposed model to the DS3 is presented in Fig. 10, which depicts the plots of the actual fault compared with the predicted faults, and the data is estimated by the MVF.

Fig. 11 represents the graph of convergence for the proposed NPSO during minimization of error generated from equation (6). It is training by failure data by DS-3 of the recurrent NN for PNSRNN model.

Table 4 shows that the proposed NPSO has much effective end-point prediction ability than the NN, PSO and DWCM. PNSRNN has the comprehensive effective end-point prediction capability than the NN, since the RE values of PSO are smallest under consideration of $E_5$. The PNSRNN based model has satisfactory end-point predictive capability compare to the models developed by NN, PSO and DWCM. It is obvious that for all end point results, the NPSO approach has better end-point prediction capability than $NN_{YP}$, $PSO_{YP}$, $NN_{YR}$, $PSO_{YR}$, $NN_{PR}$, $PSO_{PR}$, NN, PSO and DWCM.

Fig. 12 depicts the RPE curves of the proposed model have lower RPEs than the NN, PSO and DWCM. From the above figures and tables, we can conclude that NPSO approach provides a better prediction than NN, PSO and DWCM for PNSRNN model. The goodness of fit of PNSRNN model of the actual and predicted faults estimated by the MVF to all three test data (DS1, DS2 and DS3) are depicted in Figs. 3, 6, and 10 respectively. Figs. 5, 8 and 11 demonstrate the correlation between number of iteration and best fitness result for proposed PNSRNN Model.

### 2.12. Threats to validity

We have assimilated and generalized models to understand the issue in NHPP based modeling in this paper. In some cases, the fault introduction parameter in imperfect debugging is an inappropriate issue due to potentially complicated model. The proposed approach resolves most of the issues of different factors. We treated fault introduction rate by considering testing efficiency, testing team skill as additional parameter, and characteristic improvement among NHPP based SRGM modeling. The growth of introduced rate may be related to other factors such as efficiency of testing effort. In this work, we could not collect and evaluate the testing effort efficiency. The testing effort should influence the number of detected faults if test effort is not constant. Furthermore, the historical failure data sets are old enough to compare, and the scale of this system is smaller than the recent development. However, recent studies have also employed these data sets, which should protect the validity of the result of this study. We only tested proposed model with three data sets, which should protect the validity of the result of this study, but not sufficient to make uniform and/or generalized model. We also propose a two-stage Neighborhood PSO to escape the local maximal subject to fixing of issues up to a desired level.

## 3. Conclusions

This paper has presented a conceptually simple and efficient PSO algorithm within a supervised back-propagation recurrent neural network architecture. Adaptive inertia weights to train the neural network is employed by the NPSO algorithm. The algorithm is prevented from becoming trapped in local optima by the dynamic adaptive nature of the proposed NPSO. The algorithms were put into application in the context of assimilation of three well-known NHPP based SRGM for software

**Table 3**
Comparison of end point result of PNSRNN model for DS 2.

| | $NN_{YP}$ | $PSO_{YP}$ | $NN_{YR}$ | $PSO_{YR}$ | $NN_{PR}$ | $PSO_{PR}$ | $NN$ | $PSO$ | $DWCM$ | $NPSO$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $E_{10}$ | 5.6995 | 15.5255 | 47.1979 | 11.2794 | 55.5481 | 23.6493 | **0.0339** | 2.0961 | 4.4062 | 0.7035 |
| $E_9$ | 6.9240 | 21.2094 | 25.0536 | 14.9242 | 68.3383 | 17.7973 | **0.0722** | 2.2683 | 9.5636 | 0.6017 |
| $E_8$ | 7.2148 | 21.2975 | 50.5329 | 20.9679 | 89.9163 | 19.7966 | 7.7064 | 2.2317 | 8.7395 | **0.5503** |
| $E_7$ | 8.6668 | 13.5990 | 10.2012 | 20.9744 | 22.6253 | 17.5909 | 9.0823 | 2.3781 | 1.8030 | **0.4526** |
| $E_6$ | 1.9116 | 12.8895 | 20.8446 | 22.3911 | 21.9341 | 25.0868 | 9.4334 | 2.3502 | 1.4151 | **0.4233** |
| $E_5$ | 8.1315 | 23.3555 | 55.2913 | 22.9715 | 15.7956 | 35.6900 | 10.5945 | 2.7509 | 1.7351 | **0.3293** |
| $E_4$ | 1.6968 | 16.7727 | 47.4991 | 24.8672 | 29.3671 | 32.1819 | 13.4800 | 3.3984 | 1.5121 | **0.3293** |
| $E_3$ | 13.3583 | 40.9834 | 9.0755 | 26.3564 | 31.2637 | 26.2597 | 13.1002 | 3.4485 | 2.3827 | **0.2780** |
| $E_2$ | 13.4882 | 36.5031 | 33.4176 | 27.4113 | 25.7005 | 59.1024 | 13.7118 | 3.5808 | 2.1301 | **0.2679** |
| $E_1$ | 13.4968 | 13.0277 | 27.0384 | 47.0330 | 14.8235 | 12.7002 | 13.3516 | 4.5355 | 4.8166 | **0.0974** |

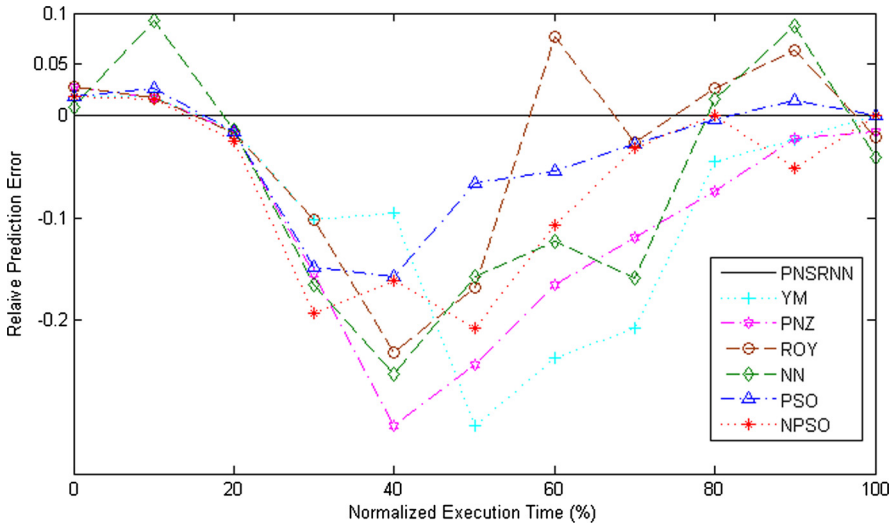


**Fig. 9.** RPE curve of PNSRNN model for DS 2.

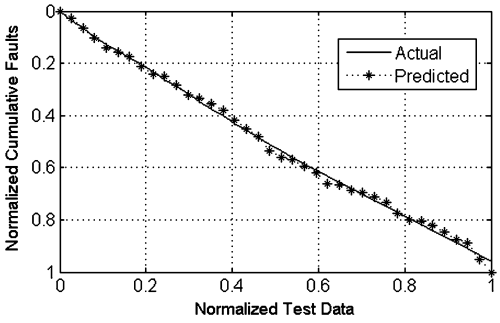


**Fig. 10.** Goodness of fit for DS 3.

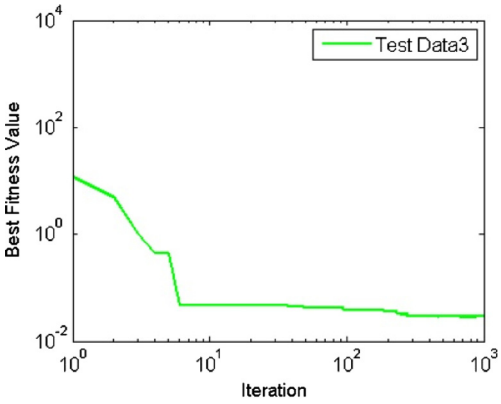


**Fig. 11.** Convergence of fitness values for DS 3.

reliability prediction. However, this approach can be applied for different applications in the many areas of engineering and science where s-shaped models are relevant. Three software failure data sets from the real time research literature were applied through the proposed NPSO algorithm. The experimental results showcased the exhibitions of better predictive quality by the NPSO than the PSO alone. The prospective of hybrid methods for prediction of software reliability with greater accuracy is thus illustrated by the combination of PSO and ANN based recurrent neural network.

The upcoming research will establish the framework of combination of NN architectures and PSO algorithms along with several other combinations of algorithms through different approaches. These different framework of combination will help for training of the failure data sets as well as for the capability of prediction of faults. Alternative soft computing approaches may be considered for the fitting and prediction under uncertainty of the flexibility logistic growth curve models along with several software reliability growth models.

## Declarations

### *Author contribution statement*

Pooja Rani, G.S. Mahapatra: Conceived and designed the experiments; Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data; Wrote the paper.

### *Funding statement*

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

**Table 4**
Comparison of end point result of PNSRNN model for DS 3.

| | $NN_{YP}$ | $PSO_{YP}$ | $NN_{YR}$ | $PSO_{YR}$ | $NN_{PR}$ | $PSO_{PR}$ | $NN$ | $PSO$ | $DWCM$ | $NPSO$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $E_{10}$ | 13.3809 | 81.6825 | 27.2272 | 46.3349 | 10.0241 | 15.0646 | 2.7035 | 2.2683 | 0.1443 | **0.0134** |
| $E_9$ | 32.1112 | 45.8041 | 51.9243 | 10.4390 | 62.6569 | 10.5335 | 7.1181 | 11.0462 | 0.1998 | **0.0137** |
| $E_8$ | 15.1254 | 40.0242 | 21.6906 | 32.9327 | 18.3568 | 21.0155 | 5.3891 | 2.2511 | 0.2019 | **0.1321** |
| $E_7$ | 86.1927 | 13.9221 | 20.3100 | 11.4459 | 10.7619 | 13.7515 | 2.5223 | 5.2331 | 0.2230 | **0.1279** |
| $E_6$ | 27.3174 | 30.5179 | 11.4434 | 47.8650 | 15.0571 | 19.3345 | 0.6116 | 8.5641 | 2.0258 | **0.1082** |
| $E_5$ | 10.8130 | 31.7522 | 15.4237 | 33.1725 | 13.3742 | 14.1085 | 7.8257 | 11.6885 | 0.2290 | **0.0472** |
| $E_4$ | 18.3703 | 18.0322 | 11.5300 | 22.6498 | 11.7402 | 12.0271 | 7.2761 | **0.0084** | 0.2383 | 0.0234 |
| $E_3$ | 23.6650 | 28.5335 | 16.6071 | 21.9761 | 17.4177 | 12.8271 | 2.7968 | 10.3322 | 0.3263 | **0.0384** |
| $E_2$ | 35.2454 | 98.8051 | 61.4583 | 31.5782 | 21.1460 | 10.3423 | 2.9338 | 3.9931 | 0.3518 | **0.3064** |
| $E_1$ | 26.7649 | 25.9798 | 29.0758 | 35.6507 | 19.8578 | 19.3188 | 9.6235 | 5.2129 | 0.3854 | **0.0675** |

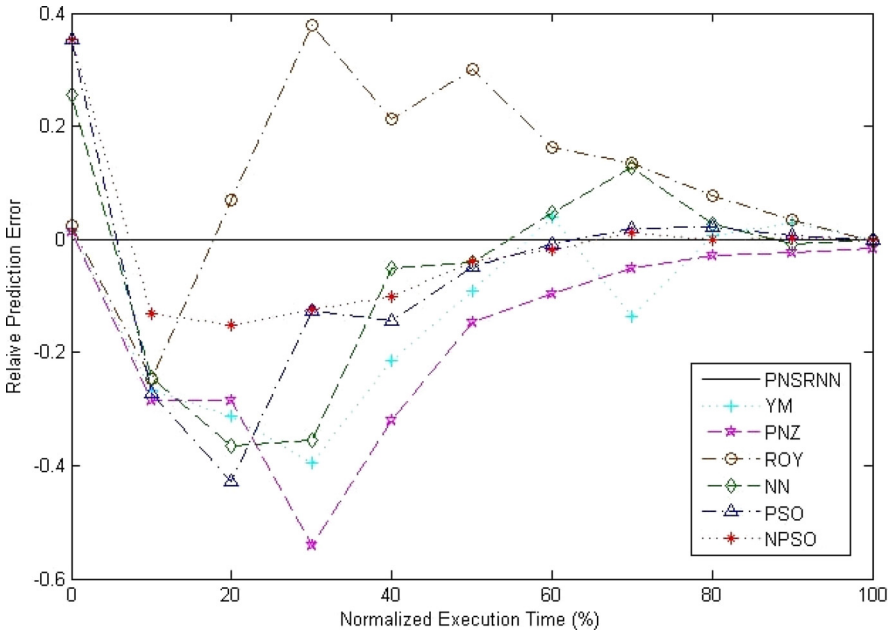


**Fig. 12.** RPE curve for DS 3 of PNSRNN model.

*Competing interest statement*

The authors declare no conflict of interest.

*Additional information*

No additional information is available for this paper.

## Acknowledgements

We are grateful to editors and reviewers for their helpful suggestions to improve this work significantly.

## References

[1] B. Littlewood, L. Strigini, Software reliability and dependability: a roadmap, in: The Future of Software Engineering, ACM Press, 2000.
[2] J.D. Musa, Software reliability engineering: more reliable software, in: Faster Development and Testing, McGraw-Hill, 2004.
[3] Michael R. Lyu, Software reliability engineering: a roadmap, in: Future of Software Engineering, 2007, pp. 55–71.
[4] ANSI/IEEE, Standard Glossary of Software Engineering Terminology, STD-729-1991, ANSI/IEEE, 1991.
[5] H. Pham, System Software Reliability, Springer, 2006.
[6] S. Yamada, S. Osaki, Software reliability growth modeling: models and applications, IEEE Trans. Softw. Eng. 11 (1985) 1431–1437.
[7] H. Pham, L. Normann, Z. Zhang, A general imperfect software debugging model with s-shaped detection rate, IEEE Trans. Reliab. 48 (2) (1999) 169–175.
[8] R. Hewett, Mining software defect data to support software testing management, Appl. Intell. 34 (2) (2011) 245–257.
[9] P. Roy, G.S. Mahapatra, K.N. Dey, An S-shaped software reliability model with imperfect debugging and improved testing learning process, Int. J. Reliab. Saf. 7 (4) (2013) 372–387.
[10] S. Chatterjee, B. Maji, A bayesian belief network based model for predicting software faults in early phase of software development process, Appl. Intell. 48 (8) (2018) 2214–2228.
[11] C.Y. Huang, M.R. Lyu, S.Y. Kuo, A unified scheme of some nonhomogeneous Poisson process models for software reliability estimation, IEEE Trans. Softw. Eng. 29 (3) (2003) 261–269.
[12] S.M. Li, Q. Yin, P. Guo, M.R. Lyu, A hierarchical mixture model for software reliability prediction, Appl. Math. Comput. 185 (2007) 1120–1130.
[13] Richard Lai, Mohit Garg, A detailed study of NHPP software reliability models, J. Softw. 7 (6) (2012) 1296–1306.
[14] Y.K. Malaiya, M.N. Li, J.M. Bieman, R. Karcich, Software reliability growth with test coverage, IEEE Trans. Reliab. 51 (2002) 420–426.
[15] Y. Liu, D. Li, L. Wang, Q. Hu, A general modeling and analysis framework for software fault detection and correction process, Softw. Test. Verif. Reliab. 26 (5) (2016) 351–365.
[16] Q.P. Hu, M. Xie, S.H. Ng, G. Levitin, Robust recurrent neural network modeling for software fault detection and correction prediction, Reliab. Eng. Syst. Saf. 92 (2007) 332–340.
[17] P. Roy, G.S. Mahapatra, K.N. Dey, An NHPP software reliability growth model with imperfect debugging and error generation, Int. J. Reliab. Qual. Saf. Eng. 21 (02) (2014) 1450008.
[18] J. Wanga, Z. Wu, Study of the nonlinear imperfect software debugging model, Reliab. Eng. Syst. Saf. 153 (2016) 180–192.
[19] S. Chandrasekharan, R.C. Panda, B.N. Swaminathan, Statistical modeling of an integrated boiler for coal fired thermal power plant, J. Heliyon 3 (6) (2017) e00322.
[20] S. Chatterjee, S. Nigam, J.B. Singh, L.N. Upadhyaya, Software fault prediction using Nonlinear Autoregressive with eXogenous Inputs (NARX) network, Appl. Intell. 37 (1) (2012) 121–129.
[21] C.Y. Huang, S.Y. Kuo, Analysis of incorporating logistic testing effort function into software reliability modeling, IEEE Trans. Reliab. 51 (3) (2002) 261–270.
[22] A.L. Goel, K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, IEEE Trans. Reliab. 28 (3) (1979) 206–211.
[23] N. Karunanithi, Y.K. Malaiya, Neural networks for software reliability engineering, in: Handbook of Software Reliability Engineering, McGraw-Hill, 1996, pp. 699–728.
[24] T.M. Khoshgoftaar, R.M. Szabo, Predicting software quality, during testing, using neural network models: a comparative study, Int. J. Reliab. Qual. Saf. Eng. 1 (1994) 303–319.
[25] L. Tian, A. Noore, Evolutionary neural network modeling for software cumulative failure time prediction, Reliab. Eng. Syst. Saf. 87 (2005) 45–51.
[26] J. Zheng, Predicting software reliability with neural network ensembles, Expert Syst. Appl. 36 (2009) 2116–2122.
[27] Y.S. Su, C.Y. Huang, Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models, J. Syst. Softw. 80 (2007) 606–615.
[28] P.K. Kapur, S.K. Khatri, M. Basirzadeh, Software reliability assessment using artificial neural network based flexible model incorporating faults of different complexity, Int. J. Reliab. Qual. Saf. Eng. 15 (2) (2008) 113–127.
[29] P.K. Kapur, D.N. Goswami, A. Bardhan, Ompal Singh, Flexible software reliability growth model with testing effort dependent learning process, Appl. Math. Model. 32 (7) (2008) 1298–1307.
[30] Q. Li, H. Pham, NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage, Appl. Math. Model. 51 (2017) 68–85.
[31] Chun-Hsiang Chuang, Zehong Cao, Yu-Kai Wang, Po-Tsang Chen, Chih-Sheng Huang, Nikhil R. Pal, Chin-Teng Lin, Dynamically weighted ensemble-based prediction system for adaptively modeling driver reaction time, IEEE Trans. Biomed. Eng. (2018), arXiv:1809.06675.
[32] P. Rani, G.S. Mahapatra, Neural network for software reliability analysis of dynamically weighted NHPP growth models with imperfect debugging, J. Softw. Test. Verif. Reliab. (2018) 15015060.

[33] R. Sitte, Comparison of software-reliability-growth predictions: neural networks vs parametric-recalibration, Trans. Reliab. 48 (3) (1999) 285–291, Griffith Univ., Brisbane, Qld., Australia.
[34] N.R. Kiran, V. Ravi, Software reliability prediction by soft computing techniques, J. Syst. Softw. 81 (4) (2008) 576–583.
[35] R. Mohanty, V. Ravi, M.R. Patra, Hybrid intelligent systems for predicting software reliability, Appl. Soft Comput. 13 (1) (2013) 189–200.
[36] T.F. Awolusia, O.L. Okea, O.O. Akinkurolerea, A.O. Sojobib, O.G. Alukoa, Performance comparison of neural network training algorithms in the modeling properties of steel fiber reinforced concrete, J. Heliyon 5 (1) (2019) e01115.
[37] G. Xu, An adaptive parameter tuning of particle swarm optimization algorithm, Appl. Math. Comput. 219 (2013) 4560–4569.
[38] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarm, in: Proceedings of the IEEE Evolutionary Computation, vol. 1, 2001, pp. 94–100.
[39] R. Poli, J. Kennedy, Particle swarm optimization, in: Time Blackwell, Swarm Intell. 1 (1) (2007) 33–57.
[40] H. Wanga, H. Sun, C. Li, S. Rahnamayan, J. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[41] C. Sun, J. Zeng, J. Pan, An improved vector particle swarm optimization for constrained optimization problems, Inf. Sci. 181 (6) (2011) 1153–1163.
[42] R. Malhotra, A. Negi, Reliability modeling using particle swarm optimization, Int. J. Syst. Assur. Eng. Manag. 4 (3) (2013) 275–283.
[43] 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).
[44] Y. Kansal, S. Choudhary, Forecasting the reliability of software via neural networks, Int. J. Comput. Sci. Inf. Technol. 5 (2) (2014) 2658–2661.
[45] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[46] Q. Bai, Analysis of particle swarm optimization algorithm, Comput. Inf. Sci. 3 (1) (2010) 180–184.
[47] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, IEEE Evol. Comput. 1 (2000) 84–88.
[48] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Inf. Sci. 209 (2012) 16–36.
[49] A.A. Mousa, M.A. El-Shorbagy, W.F. Abd-El-Wahed, Local search based hybrid particle swarm optimization algorithm for multiobjective optimization, Swarm Evol. Comput. 3 (2012) 1–14.
[50] C. Jin, S.W. Jin, Parameter optimization of software reliability growth model with S-shaped testing-effort function using improved swarm intelligent optimization, Appl. Soft Comput. 40 (2016) 283–291.
[51] V.H. Truong, S.E. Kim, Reliability-based design optimization of nonlinear inelastic trusses using improved differential evolution algorithm, Adv. Eng. Softw. 121 (2018) 59–74.
[52] M. Tian, X. Gao, Differential evolution with neighborhood-based adaptive evolution mechanism for numerical optimization, Inf. Sci. 478 (2019) 422–448.
[53] C.L. Huang, A particle based simplified swarm optimization algorithm for reliability redundancy allocation problems, Reliab. Eng. Syst. Saf. 142 (2015) 221–230.
[54] J. Wang, Z. Wu, Y. Shu, Z. Zhang, An optimized method for software reliability model based on nonhomogeneous Poisson process, Appl. Math. Model. 40 (2016) 6324–6339.
[55] P. Roy, Pooja Rani, G.S. Mahapatra, S.K. Pandey, K.N. Dey, Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. 22 (2014) 629–637.
[56] P. Roy, G.S. Mahapatra, K.N. Dey, Neuro-genetic approach on logistic model based software reliability prediction, Expert Syst. Appl. 42 (10) (2015) 4709–4718.
[57] J.C. Bansal, P.K. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in: Proc. IEEE Nature and Biologically Inspired Computing (NaBIC), Salamanca, 2011, pp. 19–21.
[58] K.Y. Cai, L. Cai, W.D. Wang, Z.Y. Yu, D. Zhang, On the neural network approach in software reliability modeling, J. Syst. Softw. 58 (2001) 47–62.
[59] Z. Xiaonan, Y. Junfeng, D. Siliang, H. Shudong, A new method on software reliability prediction, Math. Probl. Eng. (2013) 385372.
[60] Subburaj Ramasamy, C.A.S. Deiva Preetha, Dynamically weighted combination model for describing inconsistent failure data of software projects, Indian J. Sci. Technol. 9 (35) (2016).
[61] C.Y. Huang, S.Y. Kuo, Analysis of a software reliability growth model with logistic testing-effort function, in: Proc. the Eighth International Symposium on Software Reliability Engineering, 1997, pp. 378–388.
[62] P. Roy, G.S. Mahapatra, K.N. Dey, An efficient particle swarm optimization-based neural network approach for software reliability assessment, Int. J. Reliab. Qual. Saf. Eng. 24 (4) (2017) 1750019.