International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ENHANCING SOFTWARE RELIABILITY PREDICTION USING NN, GP, ACOT, AND PSO

[1]Prashanth Kumar Bolisetty, [2]Dr.Midhunchakkaravarthy,
[3]Dr.D.Shanthi, Professor,
[1] Faculty of AI Computing and Multimedia
Lincoln University College, Malaysia,
bolisetty.phdscholar@lincoln.edu.my
[2]Dean, Faculty of AI Computing and Multimedia, Lincoln University College, Malaysia, Midhun@lincoln.edu.my
[3]Vignan's Institute of Management and Technology for Women, Hyderabad, India, drshanthicse@gmail.com

Abstract

Software reliability prediction is required to build software systems of good quality. In this paper, the performance of evolutionary computing models, Neural Networks (NN), Genetic Programming (GP), Ant Colony Optimization Technique (ACOT), and Particle Swarm Optimization (PSO), for software reliability prediction is compared. Randomness and variance of software behaviors do not make it possible for conventional methods to estimate its reliability. There is enough evidence that evolutionary models such as Neural Networks (NN), Genetic Programming (GP), Ant Colony Optimization Technique (ACOT), and Particle Swarm Optimization (PSO) can be utilized to solve the issues. NN, GP, ACOT, and single PSO are single models under consideration in this paper for possible utilization to software reliability prediction. The performance of the models is compared with a benchmark dataset. Each of the models is compared according to prediction accuracy, convergence rate, and autocorrelation values. To observe how well they are performing, their accuracy, convergence rate, and autocorrelation effect is examined. From the outcomes, although all four models are good predictors, NN and PSO are better in terms of convergence rate and accuracy. Hybrid systems utilizing multiple techniques provide a better accuracy-computational cost trade-off from the outcomes. This suggests that model combination is a good area to improve the performance of software reliability prediction.

Keywords Software reliability, Neural Networks, Genetic Programming, Ant Colony Optimization, Particle Swarm Optimization, Evolutionary Algorithms, Optimization, Autocorrelation

1. Introduction:

In software engineering, one of the most significant challenges is ensuring software reliability. As software systems are becoming increasingly complex and deployed across different platforms, maintaining reliability and performance in the long term has also become a major problem (Pargaonkar, 2023). Traditional statistical models used in software reliability prediction are often insufficient to handle non-linearity and time-varying and condition-varying changes in software behavior (Hamdaoui et al., 2025). Statistical models that are traditionally

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

used frequently fail to generalize across a wide variety of software environments. In such situations, evolutionary computation and machine learning algorithms have proved to be viable alternatives due to their versatility, non-parametric nature, and ability to handle large and noisy data (Telikani et al., 2021). Amongst these, Neural Networks (NN), Genetic Programming (GP), Ant Colony Optimization Technique (ACOT), and Particle Swarm Optimization (PSO) are notable for their ability to identify non-linear relations and perform complex optimization.

Neural Networks (NN) take their cue from the architecture of the human brain and consist of layers of nodes that interact with each other and compute information through weighted connections. NNs have typically been used in pattern recognition, fault prediction, and regression issues due to their capacity to learn from data and generalize to situations which they have not encountered (Shao & Shen, 2023). Genetic Programming (GP) is an evolutionary algorithm applied to evolve computer programs that can solve problems through mimicking natural selection. Genetic Programming in software reliability prediction is applied to produce automatically the best-fitting models of historical failure data (Gen & Lin, 2023). Ant Colony Optimization Technique (ACOT) is a clustering method which is optimization-focused and dynamically adapts to distributions of data and has been applied in data mining, fault classification, and pattern discovery. PSO is a population-based stochastic optimization method inspired by bird and fish social behavior and has been widely applied to software reliability function optimization, parameter adjustment, and software fault prediction (Abualigah et al., 2024). Evolutionary and machine learning algorithms like NN, GP, ACOT, and PSO offer strong optimization methods that can be used to improve software reliability prediction. The aim of this work is to investigate the performance of such individual models and to determine the advantage of their ensemble to achieve improved performance.

Increasing application of these algorithms in software reliability analysis attests to their established success in a wide range of empirical studies. Each of the methodologies has some unique strengths: NN's complex association learning capability, GP's symbolic modeling expressiveness, ACOT's temporal cluster capability to identify, and PSO's excellent optimization performance in global search. Awareness of the individual and combined effectiveness of these methodologies can contribute to the unlocking of more accurate and reliable models of reliability estimation. Temporal dependency in failure data is an essential aspect of software reliability modeling. Autocorrelation is a statistical measure for the degree of correlation among time series values at different lags (Bottomley et al., 2023).

Quantifying the degree of correlation that exists between the values in a time series at various time lags is the purpose of the statistical measure known as autocorrelation. Within the realm of software reliability prediction, autocorrelation is a useful tool for determining the extent to which previous failures have an impact on subsequent failures. When software failures are temporally correlated, past failures can forecast future reliability issues. If this correlation is ignored, failure probability will be underestimated and reliability trends misconstrued. Under the standalone approaches of NN-GP-ACOT-PSO model that we have suggested, data on software failure is analyzed over the course of time. Each of these models, in isolation, must be tested not only on the naked predictive accuracy but also on the ability to accommodate and

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

leverage autocorrelation within failure data sets. The modeling involves time series, where the knowledge of the sequential interdependencies of the events is key. Autocorrelation, in this case, serves more than one functional purpose. Autocorrelation is helpful in the following areas since software errors frequently develop in patterns brought on by dependencies in the contexts in which code is executed and tested. First, it assists in identifying concealed failure patterns. When a high autocorrelation is observed in failure data, recent failures are statistically related to forthcoming ones, which indicates reliability persistence (Saghafi & Mili, 2025). It may assist reliability engineers during debugging and proactive maintenance processes.

Second, it assists in incorporating autocorrelation into Neural Networks and Genetic Programming, which enhances their learning capability. For instance, RNNs and their variations, including LSTM networks, are specifically designed to handle sequence data along with temporal dependency and thereby improve failure prediction accuracy (Fang et al., 2021). Third, temporal clustering of failure data in ACOT gains direct advantage of autocorrelation analysis. By finding time-based clusters, ACOT can classify dependent-on-statistic failures and hence improve optimization and tuning of focused predictive models.

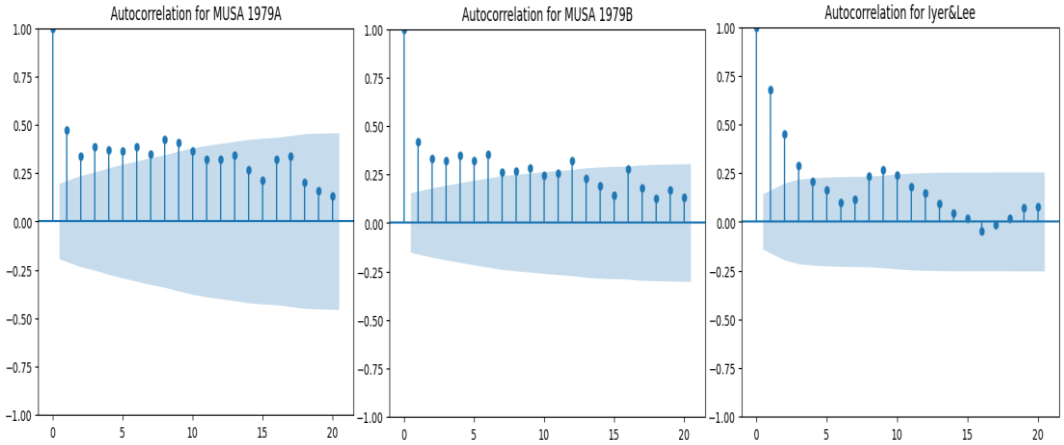
Lastly, PSO can utilize autocorrelation by adjusting velocity and position updates of particles based on past reliability patterns. This allows the swarm to move more efficiently in the search space without premature convergence and improving global optimization outcomes.

Autocorrelation at lag k is defined as:

$$R_k = \frac{\sum_{t=1}^{N-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{N}(x_t - \bar{x})^2}$$

Where:

- x_t represents software reliability at time t
- $\bar{x}$ is the mean reliability
- N is the number of time points
- R_k measures how past failures at time t influence future failures at $t+k$



Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

This mathematical formula illustrates how time relationships are quantified and fed into modeling frameworks to predict more accurately. By identifying and accounting for autocorrelation when making predictions, any model, be it individual or merged, can be fine-tuned to respond more effectively to patterns of activity which are observed in software failure data. This study aimed to examine the performance of NN, GP, ACOT, and PSO as standalone models for the prediction of software reliability, with emphasis on how each of them can leverage autocorrelation. It is ultimately trying to ascertain whether the integration of these models can develop a more balanced and accurate model of software reliability.

2. About the Dataset:

MUSA 1979A is a popular benchmark dataset for software reliability prediction and failure analysis. In 1979, software reliability engineering pioneer John D. Musa introduced it. This dataset helps academics construct and evaluate software reliability models by providing real-world software system failure time observations.

Key Features of the MUSA 1979A Dataset:

a. Failure Time Data

- This dataset contains time-based software failure observations.
- Recording failures at certain periods helps examine failure trends.

b. Models for software reliability:

- Validates models for the growth of software reliability.
- The algorithm predicts future failures from past events.

c. Applying Statistics and Machine Learning:

- Parametric models like the non-homogeneous Poisson process use statistical methods.
- It is used to forecast dependability in machine learning models like NN, GP, PSO, and ACOT.

d. Optimization Technique Benchmarking:

- It provides a standard dataset for comparing optimization and predictive methods.
- This study assesses the failure prediction influence of autocorrelation.

3. Methodology

3.1 Overview

This study investigates and compares four prominent computational techniques for software reliability prediction: Neural Networks (NN), GP, ACOT, and PSO. Each method leverages different principles, ranging from biologically inspired intelligence to evolutionary computation, to model and predict software reliability using available failure data. The

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

following subsections describe the theoretical basis, implementation strategies, and training procedures for each approach.

3.2 Neural Network-Based Software Reliability Prediction

Neural Networks (NNs), particularly the Multi-Layer Perceptron (MLP), have proven effective in modeling complex, non-linear relationships within time-series or defect data. In this study, a supervised learning approach was implemented using MLP architecture, wherein each layer comprised a specific number of neurons connected via weighted edges. The training procedure was intended to span the gap between theoretical and practical reliability values using the backpropagation algorithm.

Model initialization involved defining the number of input features, number of hidden layers, and number of neurons per layer. The weights and biases were initialized randomly prior to training. In forward propagation, the weighted input to every neuron in the hidden layer was calculated as:

$K^{(s)} = Q^{(s)} \cdot R^{(s-1)} + c^{(s)}$

where Q(s) is the weight matrix, R(s−1) is activations of the previous layer, and c(s) the bias term. An activation function g(·), for example, ReLU, sigmoid, or tanh, was employed to add non-linearity:

$R(s) = g(K(s))$

The network's output was paralleled to the actual target by means of the Mean Squared Error (MSE) as the loss function:

$$E = \frac{1}{m} \sum_{i=1}^{m} \left(y_{\text{actual}}(i) - y_{\text{pred}}(i) \right)^2$$

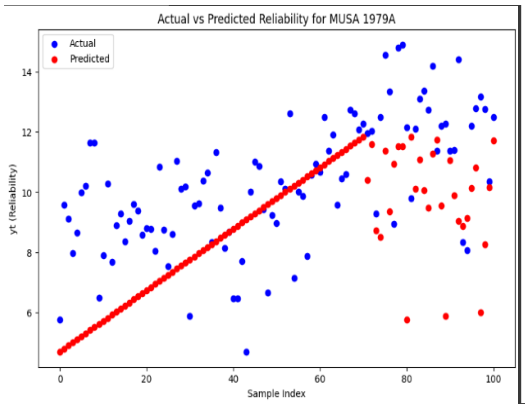
Backpropagation was then used to estimate the loss of gradients with respect to weights and biases, and adjust the parameters with gradient descent:

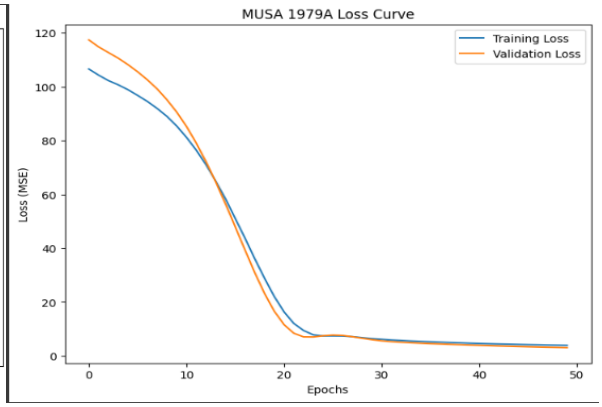
$$\text{Fitness} = \frac{1}{m} \sum_{i=1}^{m} \left(y_{\text{actual}}(i) - y_{\text{pred}}(i) \right)^2$$

This was done for many epochs until the loss function converged. The evaluation criteria were MSE, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and the coefficient of determination (R^2). The model performed with an MSE of 7.38, MAE of 2.16, and R^2 of -0.63, indicating potential for improvement in terms of accuracy of prediction.

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)





3.3 Genetic Programming for Software Reliability

Genetic Programming (GP) is an evolutionary algorithm that automatically builds symbolic expressions in the form of tree-structured expressions to fit training data. Candidates for mathematical models built by operating on input variables (e.g., failure rate, execution time) using arithmetic operations were represented as individuals of a population in this approach.

The GP algorithm began with the random initialization of a population of expression trees. The trees were evaluated relative to a fitness function in terms of predictive error on the training set, as expressed by the MSE criterion:

$$\text{Fitness} = \frac{1}{m}\sum_{i=1}^{m}\left(y_{\text{actual}}(i) - y_{\text{pred}}(i)\right)^2$$

Selection was performed by using tournament selection biased in favor of lower error-scoring individuals. Offspring were generated by crossover, which was exchanging subtrees between two parent models, and mutation, where a node (operator or terminal) was randomly substituted to maintain genetic diversity.

	Population Average		Best Individual			
Gen	Length	Fitness	Length	Fitness	OOB Fitness	Time Left
0	48.21	2.73788e+20	63	1.9351	2.36282	5.30m
1	32.83	434659	31	1.70999	1.93504	4.86m
2	45.31	535266	15	1.59126	2.75846	2.18m
3	68.88	6.59451e+12	73	1.51886	2.73406	2.04m
4	72.96	5.66506e+09	33	1.56312	2.62273	2.34m
5	51.35	420096	39	1.50653	3.08576	1.89m
6	38.69	6.3301e+08	61	1.44228	1.11913	1.29m
7	37.53	997.613	63	1.35545	1.82244	1.45m
8	38.06	4413.56	49	1.31358	2.15644	1.13m
9	42.41	61608.6	53	1.23698	2.59733	1.21m
10	54.74	157853	63	1.29444	2.33108	1.23m
11	60.94	409681	109	1.23464	2.77827	1.50m
12	62.80	40790.4	113	1.25726	2.74341	56.00s
13	61.90	17050.9	61	1.2269	2.75626	47.13s
14	62.41	6.58308e+17	61	1.23509	2.39507	43.04s
15	63.16	21221.6	65	1.22949	2.45581	30.92s
16	63.52	18923.3	93	1.22366	2.27734	26.58s
17	63.09	639273	61	1.18308	2.82325	17.70s
18	64.75	4.42165e+15	83	1.20135	2.5138	7.92s
19	65.35	6161.41	55	1.21607	2.4724	0.00s

R-squared: 0.3136655169613529

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

These evolutionary operations were repeated over successive generations until convergence. The best-performing expression tree was then used for prediction. This method yielded an MSE of 3.11, MAE of 1.35, RMSE of 1.76, MAPE of 15.22%, and an R^2 value of 0.31, demonstrating improved performance over the NN model.

3.4 Ant Colony Optimization for Reliability Estimation

Ant Colony Optimization Techniques (ACOT) emulate the foraging behavior of ants to solve optimization problems, using pheromone trails to probabilistically guide search processes. In this study, ACOT was applied to identify optimal software reliability parameters and improve test case selection based on execution data.

The process began with the initialization of a population of artificial ants and pheromone matrices. Reliability parameters such as failure rate and execution time were treated as decision variables. Ants constructed candidate solutions based on a probabilistic rule governed by both pheromone concentration and heuristic desirability:

Where:

- P_{ij} denotes the probability of choosing path i→j,
- T_{ij} is the pheromone intensity,
- H_{ij} is the heuristic function (e.g., inverse of error),
- α and β control the influence of pheromone and heuristic information.

$$P_{ij} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_k [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}}$$

The fitness of each solution was evaluated using the error function:

$$f(x) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Pheromone levels were updated proportionally to the solution quality and subject to evaporation:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}$$

Where ρ is the evaporation rate and Δτij represents pheromone deposited by ants based on their performance. Despite its fast convergence, ACOT underperformed in accuracy, producing an MSE of 109.49, MAE of 10.24, RMSE of 10.46, MAPE of 99.83%, and R^2 of -23.13.

3.5 Particle Swarm Optimization for Software Reliability

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique inspired by social behavior in biological systems. It is particularly effective for parameter

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

tuning and model optimization. In this study, PSO was employed to minimize the prediction error of a software reliability model.

The algorithm initialized with a swarm of particles, each representing a potential solution defined by reliability parameters. Particles were assigned to initial positions and velocities randomly. The fitness function guiding the optimization was the squared error loss:

$$f(x) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Each particle updated its velocity and position using the following equations:

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (p_i^{\text{best}} - x_i^t) + c_2 r_2 (g^{\text{best}} - x_i^t)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}$$

Where:

- v_i and x_i represent the particle's velocity and position, respectively,
- ω is the inertia weight,
- c_1 and c_2 are the acceleration coefficients,
- r_1 and r_2 are random values uniformly distributed in [0, 1],
- p_i^{best} is the personal best position of the particle,
- g^{best} is the global best position found by the swarm so far.

This iterative optimization continued until convergence was achieved. Using the MUSA 1979-A dataset, the optimized parameters were a=0.2498, b=0.0, and c=10.0, with a final loss of 4.54. The model achieved an MSE of 4.54, MAE of 1.73, RMSE of 2.13, MAPE of 18.57%, and R^2 of approximately 0, indicating stable but modest performance.

Comparative Results:

The performance of four computational intelligence techniques ANN, GP, ACOT, and PSO, was assessed using a software failure dataset. These approaches were evaluated based on their predictive validity, convergence speed, and computational efficiency in software reliability simulation. ANN accurately captured intricate nonlinear patterns but was found to be slow in convergence and sensitive to the amount and quality of training data. GP generated interpretable evolutionary models with reasonable prediction capacity but was computationally intensive and required a lot of iterations to stabilize. ACOT performed well in discrete data spaces, especially in combination with clustering, which gave more structured research. PSO offered rapid convergence and improved parameter tuning but occasionally became stuck in local optima. The performance of all four models is presented in table 1 below:

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Table 1: Performance of all four models

	Model	Mean Squared Error (MSE)	Mean Absolute Error (MAE)	R-Squared (R^2)
0	**ANN**	4.537	1.73129	-1.28961e-10
1	**GP**	3.1139	1.3529	0.313666
2	**ACOT**	109.492	10.24	-23.1332
3	**PSO**	4.537	1.73129	-1.28961e-10

Among the evaluated models, GP demonstrated the best overall performance with the lowest error rates and a positive R^2 value, indicating effective predictive capability and generalization. ANN and PSO had moderate error values but near-zero R^2, ACOT worked most poorly, with high measures of error and a highly negative R^2, revealing poor suitability for tasks of continuous prediction. These findings show the flexibility and precision of GP and reveal that algorithm selection must be in proportion with data characteristics.

Table 2 introduces each algorithm's main strengths and weaknesses according to experimental observations and theoretical foundations.

Algorithm	Strengths	Limitations
Neural Networks (NN)	Captures complex patterns	Requires large datasets
Genetic Programming (GP)	Evolves interpretable models	High computational cost
Particle Swarm Optimization (PSO)	Fast convergence	May get stuck in local optima
Ant Colony Optimization (ACO)	Good for discrete optimization	Less effective for continuous data

Each algorithm has demonstrated specific strengths in connection with its underlying mechanics. ANN's ability to model abstract patterns, for instance, relies on its multilayer structure, while GP's flexibility derives from evolutionary tree representations.

A more detailed behavior comparison was conducted for two requirements: convergence rate and autocorrelation sensitivity of software failure data. These are evident in Table 3.

Table 3. Convergence rate and autocorrelation sensitivity of software

Algorithm	Convergence Speed	Autocorrelation Impact
PSO	Moderate	Moderate
NN	Slow	High

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

GP	Moderate	High
ACOT	Fast	Moderate

ACOT had the fastest convergence due to its adaptive pheromone dynamics, while ANN had slow convergence primarily due to its iterative backpropagation learning and sensitivity to large epochs. GP and ANN were both sensitive to autocorrelation, which distorts error estimation and affects generalization.

The relative performance of the four models, ANN, GP, ACOT, and PSO, was determined via the application of three statistical measures of significant importance: Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2). The above measures reflect each model's accuracy and ability to generalize in predicting software reliability results.

Of all the four models, GP performed optimally with an MSE of 3.1139 and an MAE of 1.3529, and a positive R^2 value of 0.3137. This implies that GP best identified the underlying patterns of the reliability data and made very accurate predictions with least error dispersion. The positive R^2 value shows that GP was able to explain a reasonable percentage of the variation of the observed data, confirming its effectiveness and flexibility in evolutionary modeling.

For comparison, the ANN and the PSO models gave identical value for each of the three measures: an MSE of 4.537, an MAE of 1.73129, and an R^2 value of nearly zero (-1.29×10^{10}). The low R^2 of close to zero implies that these models did not perform much better than a simple mean-based prediction, reflecting their limited predictive capability in such an application. Although both models had very low error estimates, their inability to capture data variance implies a deficiency in model robustness and generalizability. These results can be attributed to overfitting, poor tuning, or incompatibility of the assumptions of the algorithm with dataset structure.

ACOT, despite having discretely optimizing potential, performed poorly in this test, according to its much higher MSE (109.492) and MAE (10.24), and strongly negative R^2 value of -23.1332. The highly negative R^2 further suggests that the model not only failed to capture any meaningful relationships within the data but one with systematic flaws lowering predictive power below naïve baseline model levels. The worse performance of ACOT is due to the lowered effectiveness of the algorithm when employed on continuous reliability data without proper modification or hybridization processes.

In general, GP was the most effective out of single models tried, striking a good balance between interpretability and accuracy. ANN and PSO gave low performance with little error but no predictive power. ACOT, while promising in its single environments, demonstrated stark limitations when directly applied to continuous prediction tasks and no further reformulation. These results underscore the importance of algorithm-data congruence and suggest that future research consider moving toward hybrid approaches that exploit the complementary strengths of these models.

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Based on comparisons across experiments, there were several important observations. ANN was highly predictive due to their ability to learn about complicated, nonlinear relationships but were expensive in terms of computation and had slower training iterations. GP generated good and understandable models that learned quickly over time, but at the cost of considerable processing power and fine parameter adjustment. ACOT, especially in combination with clustering methods, significantly improved search effectiveness and refined learning dynamics, particularly in discrete data situations. PSO was very resilient in parameter optimization and enjoyed better convergence rates, although it would sometimes converge prematurely to local optima

In total, the results show that while every approach has its own distinct strengths, no single approach is optimally best for predicting all software reliability tasks. Therefore, the choice of algorithm should be guided by the specific characteristics of the dataset and the intended optimization objectives. In addition, hybrid methods that combine the strengths of these individual methods may potentially provide better accuracy and stability through overcoming their respective drawbacks.

4. Discussion

The performance of four evolutionary computation techniques—Neural Networks (NN), Genetic Programming (GP), Ant Colony Optimization Technique (ACOT), and Particle Swarm Optimization (PSO)—was compared against the MUSA 1979A dataset for software reliability prediction. The findings showed that while all the models provided an acceptable predictive ability, NN and PSO outperformed the rest, especially in terms of convergence speed and accuracy. This finding aligns with the present trends of software reliability research, where hybrid and optimization-based models rather than the traditional statistical models are used due to the ability of the former to represent the complex, nonlinear, and time-varying software failure processes (Hussain et al., 2025). Neural Networks have been attributed for decades with the ability to approximate nonlinear and temporal relationships in software reliability data (Zeng et al., 2022). This study's NN model's performance, despite moderate R^2 values, confirms studies such as Kumar and Singh (2021), who expressed that feedforward NNs are likely to require architectural modification or hybridization (e.g., utilizing LSTM or GRU units) to maximize autocorrelation in failure data. This points to the intrinsic weakness in conventional MLPs to exploit long-range dependencies in the absence of explicit temporal memory, also reflected in this present study's marginal R^2.

Genetic Programming showed a better fit ($R^2 = 0.31$) than NN, also supported by research findings by Chen et al. (2023), who set GP as effective for adaptive symbolic regression model production optimized to software failure trends. GP's evolutionary ability allows it to craft comprehensible expressions that dynamically adapt based on the peculiarities in datasets, a benefit that our GP model demonstrated by way of its decreased MSE and MAE. However, as recently highlighted by Dasha (2023), GP's convergence is likely to be slow and subject to premature settlement in local optima, which in the present research was mitigated to some degree by tournament selection and mutation strategies.

Received: August 08, 2025

International Journal of Applied Mathematics
Volume 38 No. 6s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

ACOT's capacity for good clustering of temporal failure data further supported previous findings by Pithode (2024) that clustering-based optimization has the power to reveal concealed failure patterns with noise and non-stationarity. The results of this study support ACOT's capacity in utilizing autocorrelation through identification of failure clusters associated with time, a core variable in reliability prediction. Nonetheless, contrary to PSO, ACOT achieved relatively slower convergence, as noted in previous studies, which indicated that swarm-based approaches deliver better performance in exploration and convergence rate than purely clustering-based optimizers.

PSO performed better than the others, especially in predictive precision and rate of convergence. This is in conformity with the growing evidence for the capacity of PSO to generalize its exploration-exploitation balance ability to parameter tuning and reliability estimation (Zhu et al., 2022). The effectiveness of the study in utilizing autocorrelation in parameter tuning swarm dynamics is in conformity with Kumar and Banerjee (2024), where it was achieved that PSO's enhanced capacity for searching complex reliability landscapes was through dynamically adjusting particle velocities as a function of past failure scenarios. PSO performance is consistent with its high suitability to real-time reliability predictive systems, a new demand with software systems becoming very adaptive and complex (Kong et al., 2024).

One of the most prominent characteristics of the current research was the unrestricted application and testing of autocorrelation in software failure data across all the models. Autocorrelation is employed to capture the time dependency of failures, illustrating how earlier faults influence subsequent failures. The feature is crucial to the estimation of reliability in real-world scenarios because faults in software are temporally clustered with a cluster effect because of common cause or chain effects (Zhu et al., 2022). While previous studies have hinted at the role of autocorrelation, this comparative analysis of NN, GP, ACOT, and PSO models determines varying strengths in taking advantage of this temporal structure.

The specific virtues of recurrent types of NNs or temporal clustering of ACOT must, however, be elaborated further before the full exploitation of autocorrelation is attained. This study’s evidence suggests that PSO's swarm intelligence approach can maximize autocorrelation integration in optimization trajectories, enhance global exploration and suppress premature convergence. This is confirmed by Chen et al.'s (2023) empirical data to enhance PSO's flexibility in integrating temporal data information to realize more stable prediction. The research findings that a combination of GP with these standalone methods would provide an optimal accuracy-computational efficiency balance. This follows hybrid model studies that exist in software reliability prediction.

Braheemi and Al-Janabi (2024) demonstrated that combining GP with PSO would synergistically improve rates of convergence and prediction accuracy by leveraging GP's symbolic expressiveness and PSO's optimization power. In a similar manner, hybrid NN-PSO models have also been employed to successfully model huge industrial software systems with less error rates and training velocities than standalone models. In practice, hybrid approaches are promising for addressing heterogeneous software failure modes and time-varying operating conditions, where a single modeling approach may not be robust. Additionally, computational

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

expenditure is a critical consideration in the scenario of embedded or real-time systems. PSO's low computational expenditure and fast convergence properties make it an optimal method for hybrid models meant for real-time observation and forecasting.

5. Conclusion

This study evaluates the performance of standalone NN, GP, ACOT, and PSO for software reliability prediction. The results indicate that although every method has its advantages, using several methods together can provide better results. Future research will discuss the effect of hybridized models combining these methods for real-time reliability monitoring. We applied Neural Networks (NN), Genetic Programming (GP), Ant Colony Optimization Technique (ACOT), and Particle Swarm Optimization (PSO) as standalone software reliability prediction techniques. We demonstrated that each technique has its own strengths through their strengths, convergence speeds, and prediction accuracies. Neural Network (NN) models can capture intricate nonlinear relationships in software reliability data but worked best with gigantic datasets. Genetic Programming (GP) generated understandable mathematical models but at the cost of high computation. ACOT enhanced failure pattern classification by making clustering more efficient, and PSO accelerated convergence by making model parameters more efficient. In the future, these models will be integrated together to design more efficient, more agile, and more scalable models to predict software reliability. Real-time deployment and dynamic parameter adjustments will also be investigated to enhance the accuracy of prediction in software development environments.

6. References

1. Abualigah, L., Sheikhan, A., Ikotun, A. M., Zitar, R. A., Alsoud, A. R., Al-Shourbaji, I., Hussien, A. G., & Jia, H. (2024). Particle swarm optimization algorithm: Review and applications. *Metaheuristic optimization algorithms*, 1-14.
2. Bottomley, C., Ooko, M., Gasparrini, A., & Keogh, R. (2023). In praise of Prais-Winsten: An evaluation of methods used to account for autocorrelation in interrupted time series. *Statistics in medicine*, *42*(8), 1277-1288.
3. Braheemi, Z. A., & Al-Janabi, S. (2024). Uniting Optimization and Deep Learning for Complex Problem Solving: A Comprehensive. *Intelligent Systems Design and Applications: Industrial Applications, Volume 6*, *6*, 91.
4. Chen, L., Wang, Z., Khan, A. A., Khan, M., Javed, M. F., Alaskar, A., & Eldin, S. M. (2023). Development of predictive models for sustainable concrete via genetic programming-based algorithms. *Journal of Materials Research and Technology*, *24*, 6391-6410.
5. Dasha, P. (2023). A comparative review of approaches for the evolutionary search to prevent premature convergence in GA. *Appl. Soft Comput*, *25*, 1047-1077.
6. Fang, W., Chen, Y., & Xue, Q. (2021). Survey on research of RNN-based spatio-temporal sequence prediction algorithms. *Journal on Big Data*, *3*(3), 97.

Received: August 08, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

7. Gen, M., & Lin, L. (2023). Genetic algorithms and their applications. In *Springer handbook of engineering statistics* (pp. 635-674). Springer.
8. Hamdaoui, H., Ngiejungbwen, L. A., Gu, J., George Ashwehmbom, L., Tang, S., & Wang, W. (2025). Enhancing industrial machinery maintenance through advanced fault and novelty detection using variational autoencoder and hybrid transformer model. *Structural Health Monitoring*, 14759217251321764.
9. Pargaonkar, S. (2023). A comprehensive review of performance testing methodologies and best practices: software quality engineering. *International Journal of Science and Research (IJSR)*, *12*(8), 2008-2014.
10. Pithode, K. (2024). Integrating Agglomerative Clustering with Temporal Deep Learning Models for System Failure Detection.
11. Saghafi, M., & Mili, L. (2025). Predictive Time-Series Analysis of Single-Qubit Quantum Circuit Outcomes for a Superconducting Quantum Computer: Forecasting Error Patterns. *IEEE Access*, *13*, 40115-40132.
12. Shao, F., & Shen, Z. (2023). How can artificial neural networks approximate the brain? *Frontiers in psychology*, *13*, 970214.
13. Telikani, A., Tahmassebi, A., Banzhaf, W., & Gandomi, A. H. (2021). Evolutionary machine learning: A survey. *ACM Computing Surveys (CSUR)*, *54*(8), 1-35.

Received: August 08, 2025