Adaptive Crow Search Optimization With Anfis For Predicting The Software Reliability In Banking Sector

P. Lakshminarayana , Dr T V SureshKumar

Abstract: Reliability of a system has a direct impact on the success of any software system. Predicting the reliability helps to optimize the maintenance of the software. Rapid changes in hardware and software technologies lead to inventions of new methodologies and needs developing and validating a reliability predicting model for each method. To solve this problem, we have presented an adaptive technique which combines crow search optimization technique and Adaptive NeuroFuzzy Inference System. The chosen data set is given as the input of proposed method and estimate the reliability. To increase the convergence rate and reliability prediction rate, the parameters of ANFIS are optimized using CSO algorithm.The proposed technique is implemented in the platform of MATLAB and their corresponding performance is compared with the conventional techniques such as ABC-ANFIS, ANT-ANFIS, GA-ANFIS, PSO-ANFIS and ANFIS.

Keywords: Reliability Prediction, Crow Search Optimization, Adaptive Neuro Fuzzy Inference System, Root Mean Square Error

1. INTRODUCTION

Software is an integral part of many critical and non-critical applications, and virtually any industry is dependent on computers for their basic functioning. As computer software permeates our modern society, and will continue to do so at an increasing pace in the future, the assurance of its quality becomes an issue of critical concern [1-3]. Normally, the software development life cycle includes a number of phases such as requirement analysis, design, construction, testing, deployment and maintenance, usually supported by other activities such as configuration management, engineering management and quality assurance [4-7]. One of the most important issues encountered during the process of software development is a company's ability to accurately estimate the efforts and necessary costs of the initial stage of development [8]. Predicting the effort required to create software has been based on numerous software size, models and functions [9]. Prediction is needed during the initial phase of software project developments. Also, software effort estimation is necessarily required at the inception phase in order to bid for a software project and assign human resources [10].

Software reliability prediction plays a very important role in the analysis of software quality and balance of software cost. Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment. Its evaluation includes two types of activities: reliability estimation and reliability prediction [11-13]. However, predicting the variability of software reliability with time is very difficult. Techniques to measure and ensure reliability of hardware have seen rapid advances, leaving software as the bottleneck in achieving overall system reliability. Incorrect software estimation leads to late delivery, surpassing the budget and project failures.

- *P. Lakshminarayana, PhD Scholar, B.M.S.College of Engineering Email: phdresearchpaper19@outlook.com*
- *Dr T V SureshKumar is Professor of MCA, BangaloreM S R Institute of Technology, Bangalore*

According to the International Society of Parametric Analysis (ISPA) and the Standish Group International, the main reasons behind project failures include optimism in conducting software estimation as well as misunderstanding and uncertainty in software requirements [14-15]. At the inception of each software project, project managers use several techniques to predict software size and effort that will help them learn the cost, required time and the number of staff required to develop a project. Software fault prediction (SFP) is typically used to predict faults in software components. During the early phase of software life cycle, the total number of software faults is estimated and reduced the faults in the software. Numerous techniques are utilized to predict and estimate the software reliability such as Fuzzy Logic Controller, Neural Network (NN), Genetic Algorithm (GA), COCOMO, SLIM and SEER-SEM regression models [16-19 The rest of the document follows as: Section 2 mentioned about the literature review of the recent articles and the section 3 described about the proposed methodology. The performance analysis of the proposed and existing methods is presented in the section 4. Finally the conclusion is presented in the section 5.

2. LITERATURE SURVEY

A lot of works are presented to predict and estimate the software reliability; some of them are reviewed here, Cong Jin et al. [20] carried out an estimation of distribution algorithms (EDA) in order to maintain the diversity of the population, and then a hybrid improved estimation of distribution algorithms (IEDA) and SVR model, called IEDA-SVR model. IEDA was used to optimize parameters of SVR, and IEDA-SVR model was used to predict software reliability. They compared IEDA-SVR model with other software reliability models using real software failure datasets. ArunimaJaiswalet al. [21] demonstrated the ML techniques for software reliability prediction based on selected performance criteria. They were applied ML techniques including adaptive neuro fuzzy inference system (ANFIS), feed forward back propagation neural network, general regression neural network, support vector machines, multilayer perceptron, Bagging, cascading forward back propagation neural network, instance based learning, linear regression, M5P, reduced error pruning

tree, M5Rules to predict the software reliability on various datasets being chosen from industrial software. Based on the experiments conducted, it was observed that ANFIS yields better results in all the cases and thus can be used for predicting software reliability since it predicts the reliability more accurately and precisely as compared to all other above mentioned technique. Subhashis Chatterjee et al. [22] advocated a fuzzy rule-based generation algorithm in interval type-2 fuzzy logic system for fault prediction in the early phase of software development. Their proposed model considers reliability relevant software metric and earlier project data as model inputs. Type-2fuzzy sets have been used to reduce uncertainties in the vague linguistic values of the software metrics. A rule formation algorithm has been developed to overcome inconsistency in the consequent parts of large number of rules. Twenty-six software project data help to validate the model, and a comparison has been provided to analyze the proposed model's performance. Mengmeng Zhu et al. [23] carried out a non-homogeneous Poisson process (NHPP) software reliability model with a pioneering idea by considering software fault dependency and imperfect fault removal. Two types of software faults were defined, Type I (independent) fault and Type II (dependent) fault, according to fault dependency. Two phases debugging processes, Phase I and Phase II, were proposed according to the debugged software fault type. A small portion of software faults that software testers were not able to remove was also considered in both phases in the proposed model. The illustration of the model effectiveness was based on the three datasets collected from industries. ChanderDiwakeret al. [24] advocated a Bio Inspired Soft Computing Techniques for predicting software reliability. Their paper focused on assessment of commonly used soft computing technique like Genetic Algorithm (GA), Neural-Network (NN), Fuzzy Logic, Support Vector Machine (SVM), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony (ABC). They represent the working of soft computing techniques and assessment of soft computing techniques to predict reliability. The parameter considered while estimating and prediction of reliability were also discussed. Their study could be used in estimation and prediction of the reliability of various instruments used in the medical system, software engineering, computer engineering and mechanical engineering also. These concepts can be applied to both software and hardware, to predict the reliability using CBSE. NeelamdhabPadhyet al. [25] carried out an evolutionary computing-based artificial intelligence or machine learning scheme for regression tests to be used for reusability estimation. Their model was popularly called an aging-resilient software reusability forecast representation. Their proposed system employs predominant object-oriented software metrics, such as Chidamber and Kemerer's metrics to examine reusability. Their primary constructions have been developed for estimating the metrics from the UML/class diagrams. It was feasible to derive an efficient and robust reusability prediction model for web-service products using object-oriented metrics. TirimulaRaoBenalaet al. [26] have investigated the effectiveness of differential evolution (DE) algorithm, for optimizing the feature weights of similarity functions of ABE by applying five successful mutation strategies. They found improvements in predictive performance of our DABE technique over ABE, particle swarm optimization-based feature weight optimization in ABE, genetic algorithm-based feature weight optimization in ABE, self-adaptive DE-based feature weight optimization ABE, adaptive differential evolution with optional external archive-based feature weight optimization ABE, functional link artificial neural network, artificial neural network with back propagation learning based software development effort estimation (SDEE), and radial basis function-based SDEE. AndrésGarcía-Florianoet al. [27] advocated software metrics algorithms and their primary constructions for estimating the metrics from the UML/class diagrams. It was also found that OO-CK metrics, particularly complexity, cohesion and coupling-related metrics can be helpful in predicting reusability in web-service software products. Considering the above-mentioned key contributions, it can be stated that the proposed research could be of paramount significance in next-generation software computation systems, primarily for software component reusability, reliability, survivability, aging prediction and stability, and for software excellence assurance purposes. Hamza Turabiehet al. [28] advocated a FS approach to enhance the performance of a layered recurrent neural network (L-RNN), which was used as a classification technique for the SFP problem. Three different wrapper FS algorithms (i.e., Binary Genetic Algorithm (BGA), Binary Particle Swarm Optimization (BPSO), and Binary Ant Colony Optimization (BACO)) were employed iteratively. To assess the performance of their proposed approach, 19 real-world software projects from PROMISE repository were investigated and the experimental results were discussed. Receiver operating characteristic – area under the curve (ROC-AUC) was used as a performance measure.

3. Proposed Software Reliability Testing Technique using ACSO-ANFIS

Software reliability plays a key role in software quality. In order to improve accuracy and consistency of software reliability prediction, in this paper we have proposed the applicability of Adaptive Crow Search Optimization with Adaptive Neuro Fuzzy Inference System (ACSO-ANFIS) as a model to predict the number of faults in the software during development and testing processes. The model has been applied on data sets collected across several standard software projects during system testing phase with fault removal. Banking sector data is used for the testing process. A comparative study among the proposed CSO-ANFIS with some traditional parametric software reliability growth model's performance is carried out. The architecture of the proposed technique is given in Fig.1.

Let us consider the database $D = \{ p(\vec{x}_i) \in i = 1,2,..., n$ used for training the ANFIS to predict the reliability.

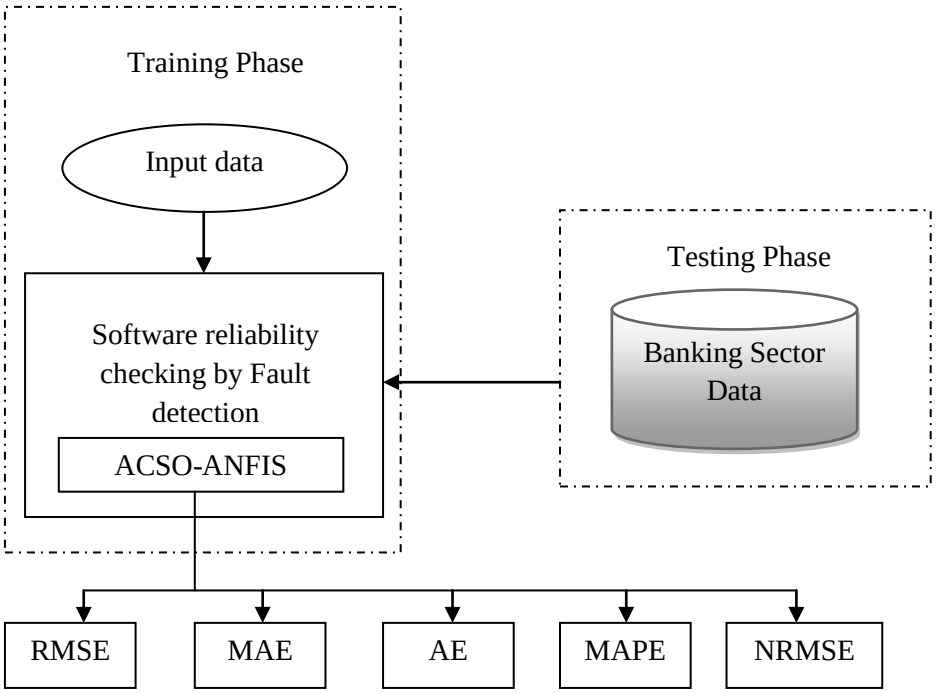

Fig.1: *Architecture of the proposed software reliability testing technique*

3.1 Adaptive NeuroFuzzy Inference System (ANFIS)

ANFIS is a hybrid approach developed by combining Fuzzy Inference System that has a high level of reasoning capability, which is made more robust using the low-level computational power of a neural network. The fuzzy logic which is a multi-defined logic defines the Intermediate value between true and false. Thus it can range from very low, low, high, to very high depending on the scale considered. Using the number of fuzzy rules (IF-THEN), a knowledge base is formed in a fuzzy reasoning system.

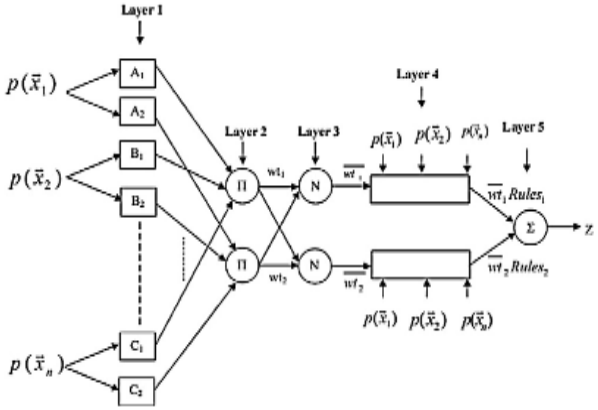

Fig.2: *Architecture of ANFIS*

The input $p(\vec{x}_1), p(\vec{x}_2),..., p(\vec{x}_n))$ from the database are passed through the well known classifier named ANFIS to detect faults which comprises five layers of nodes.Out of five layers, the first and the fourth layers possess adaptive nodes whereasthesecond,thirdandfifthlayerspossessfixednodes.TheearchitectureoftheANFISisgivenin Fig.2.

The Rule basis of the ANFIS is of the form:

If $p(\vec{x}_1)$ *is* A_i, $p(\vec{x}_2)$ *is* B_i *then* C_i *is*

$$R_i = a_i\, p(\vec{x}_1) + b_i\, p(\vec{x}_2) + c_i\, p(\vec{x}_n) + f_i \quad (1)$$

Where $p(\vec{x}_1), p(\vec{x}_2),..., p(\vec{x}_n)$ are the inputs, A_i, B_i and C_i are the fuzzy sets, R_i is the output within the fuzzy region specified by the fuzzy rule, a_i, b_i, c_i and f_i are the design parameters that are determined by the training process.

Layer-1: Every node i in this layer is a square node with a nodefunction.

$$O_{1,i} = \mu_{A_1} p(\vec{x}_1),\ O_{1,i} = \mu_{B_1} p(\vec{x}_2),\ O_{1,i} = \mu_{C_1} p(\vec{x}_n) \quad (2)$$

Usually $\mu_{A_1} p(\vec{x}_1)$, $\mu_{B_1} p(\vec{x}_2)$, $\mu_{C_1} p(\vec{x}_n)$ are chosen to be bell-shaped with maximum equal to 1 and minimum equal to 0 and are defined as

$$\mu_{A_1} p(\vec{x}_1) = \mu_{B_1} p(\vec{x}_2) = \mu_{C_1} p(\vec{x}_n) = \left(\frac{1}{1 + \left(\frac{x - o_i}{p_i} \right)^{2q_i}} \right) \quad (3)$$

Where o_i, p_i, q_i is the parameter set. These parameters in this layer are referred to as premise parameters.
Layer 2: Every node in this layer is a circle node labeled which multiplies the incoming signals and sends the product out. For instance,

$$O_{2,i} = wt_i = \mu_{A_i} p(\vec{x}_1) \times \mu_{B_i} p(\vec{x}_2) \times \mu_{Ci} p(\vec{x}_n), i = 1,2 \quad (4)$$

Each node output represents the firing strength of a rule.
Layer 3: Every node in this layer is a circle node labeled N. The i[th] node calculates the ratio of the i[th] rules firing strength to the sum of all rule's firing strengths:

$$O_{3,i} = wt_i = \frac{wt_i}{(wt_1 + wt_2)}, i = 1,2 \quad (5)$$

Layer-4: Every node i in this layer is a square node with a node function

$$O_{4,i} = wt_i R_i, i = 1,2 \quad (6)$$

Where wt_i is the output of layer-3 and a_i, b_i, c_i and f_i are the parameter set. Parameters in this layer will be referred to as consequent parameters.
Layer5: The single node in this layer is a circle node labeled thatcomputestheoveralloutputasthesummationofalli ncomingsignals.

$$O_{5,i} = \frac{\sum_i wt_i R_i}{\sum_i wt_i} \quad (7)$$

Then the predefined threshold value ω and the result of the neural network (Z) are compared which is given in the following equation:

$$result = \begin{cases} fault, & Z < \omega \\ nofault, & Z \geq \omega \end{cases} \quad (8)$$

Based on the comparison fault is detected in the software. In order to improve the predictive accuracy of ANFIS and avoid falling in to the local optimum, parameter learning is done by Crow search algorithm. Parameter learning process of ANFIS can be summarized as the adjusting of premise parameters and consequent parameters, Crow search algorithm is used to train the parameters which includes a forward channel and a reverse channel. In forward channel, the premise parameters are fixed and the input signs transfer through the calculation of each layer5 and the consequent parameters are identified by least square method, then the consequent parameters are fixed in reverse channel, error signs are transferred to layer 2 to update premise parameters.

3.2 Crow Search Optimization Algorithm

Randomly initialize the position of a flock of N crows in the search space
Evaluate the position of the crows
Initialize the memory of each crow
While run <maxruns
while t <itmax
for A = 1:N
randomly choose one of the crows to follow
define an awareness probability
ifrb>=apbt calculate new position using (5)
else xa,t+1 = a random position of search space
endif
end for
check the feasibility of new positions
evaluate the new position of the crows
Update the memory of crows
end while

***Figure.3**. General Pseudo code of Crow Search Algorithm*

Crows are considered the most intelligent birds. They contain the largest brain relative to their body size. Based on a brain-to-body ratio, their brain is slightly lower than a human brain. Evidences of the cleverness of crows are plentiful. They have demonstrated self-awareness in mirror tests and have tool-making ability. Crows can remember faces and warn each other when an unfriendly one approaches. Moreover, they can use tools, communicate in sophisticated ways and recall their food's hiding place up to several months later. Crows have been known to watch other birds, observe where the other birds hide their food, and steal it once the owner leaves. If a crow has committed thievery, it will take extra precautions such as moving hiding places to avoid being a future victim. In fact, they use their own experience of having been a thief to predict the behavior of a pilferer, and can determine the safest course to protect their caches from being pilfered.
The principles of CSA are listed as follows:

- Crows live in the form of flock.
- Crows memorize the position of their hiding places.
- Crows follow each other to do thievery.
- Crows protect their caches from being pilfered by a probability

It is assumed that there is an N -dimensional environmentincluding a number of crows. The number of crows is S and the position of crow i at time iter in thesearch space is specified by a vector $c^{i,iter}\ (i \in \{i,2,...,\ S;iter\ \in 1,2,..,\ iter_{\max}\})$ where $c^{i,iter} = \{c_1^{i,iter}, c_2^{i,iter}, ..., c_s^{i,iter}\}$ and $iter_{\max}$ is the maximum number of iterations. Each crow has amemory in which the position of its hiding place is memorized.At iteration iter, the position of hiding place of crow i is shownby $l^{i,iter}$. This is the best position that crow i has obtained so far.Indeed, in memory of each crow the position of its best experiencehas been memorized. Crows move in the environment and searchfor better food sources (hiding places).

The step-wise procedure for the implementation of CSA is given in this section.

Step 1: Initialize problem and adjustable parameters:

The optimization problem, decision variables and constraints are defined. Then, the adjustable parameters of CSA (flock size(S), maximum number of iterations ($iter_{\max}$), flight length (ftl)and awareness probability (AP)) are valued.

Step 2: Initialize position and memory of crows:

S -crows are randomly positioned in an N -dimensional search space as the members of the flock. Each crow denotes a feasible solution of the problem and n is the number of decision variables.

$$C = \begin{bmatrix} c_1^1 & c_2^1 & \dots & c_n^1 \\ c_1^2 & c_2^2 & \dots & c_n^2 \\ . & . & \dots & . \\ . & . & \dots & . \\ c_S^1 & c_S^2 & \dots & c_S^n \end{bmatrix} \quad (9)$$

The memory of each crow is initialized. Since at the initial iteration, the crows have no experiences, it is assumed that they havehidden their foods at their initial positions.

$$M = \begin{bmatrix} m_1^1 & m_2^1 & \dots & m_n^1 \\ m_1^2 & m_2^2 & \dots & m_n^2 \\ . & . & \dots & . \\ . & . & \dots & . \\ c_1^N & c_2^N & \dots & c_n^N \end{bmatrix} \quad (10)$$

Step 3: Evaluate fitness (objective) function:

For each crow, the quality of its position is computed by inserting the decision variable values into the objective function, as it is the optimization of prediction effect in the ANFIS, the reciprocal of root mean square (RMSE) between the actual output and predictive output is selected as fitness function, RMSE can be expresses as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{S} (Z - result)^2}{n}} \quad (11)$$

Where, Z is the actual output and result is the prediction output, the purpose of optimization is to get the smallest error between actual output, the maximum and average value of all individuals will be calculated after each generation.

Step 4: Generate new position:

Crows generate new position in the search space as follows: suppose crow i wants to generate a new position. For this aim, this crow randomly selects one of the flock crows (for example crow j) and follows it to discover the position of the foods hidden by this crow (m_j). The new position of crow i is obtained by Eq.(10). This process is repeated for all the crows.

Step 5:Check the feasibility of new positions:

The feasibility of the new position of each crow is checked. If thenew position of a crow is feasible, the crow updates its position. Otherwise, the crow stays in the current position and does not move to the generated new position.

Step 6: Evaluate fitness function of new positions:

The fitness function value for the new position of each crow is computed.

Step 7: Update memory:

The crows update their memory as follows:

$$m^i = \begin{cases} c^{i,iter+1} & f\left(c^{i,iter+1}\right) \ is \ better \ than \ f\left(m^{i,iter}\right) \\ m^{i,iter} & o.w. \end{cases} \quad (12)$$

Where $f(.)$ denotes the objective function value. It is seen that if the fitness function value of the new position of a crow is better than the fitness function value of the memorized position, the crow updates its memory by the new position. Thus the optimal parameters can be found and the possibility of falling into local minimum can be avoided.

Step 8: Check termination criterion

Steps 4–7 are repeated until $iter_{\max}$ is reached. When the termination criterion is met, the best position of the memory in terms of the objective function value is reported as the solution of the optimization problem.

To evaluate the proposed technique, Root Mean Square Error (RMSE), Average Error (AE), Mean Absolute Error (MAE), Mean absolute percent error (MAPE) and Normalized Mean Square Error (NRMSE) are calculated.

4. EXPERIMENTAL RESULTS AND DISCUSSION

An adaptive technique which combined CSO and ANFIS technique has been proposed to evaluate the software reliability in this paper. The proposed technique is implemented in the platform of MATLAB. The performance of the proposed technique is evaluated and compared with the conventional techniques such as ANFIS combined with Artificial Bee Colony Algorithm (ABC), Particle Swarm Optimization (PSO) algorithm, Genetic Algorithm (GA) and ANT Algorithm respectively.

Table.1 shows the comparison of reliability values obtained against faults. The reliability prediction calculated for ten numbers of components. The table shows the faults against number of component and the predicted reliability for proposed CSO-ANFIS, ABC-ANFIS, PSO -ANFIS, GA-ANFIS, ANT-ANFIS and ANFIS techniques. On looking at the table, ANFIS without optimization algorithm has lower predictability rate and the proposed ANFIS-CSO has highest predictability rate. We can arrange the performance of the techniques in descending order as proposed CSO-ANFIS > ABC-ANFIS > PSO-ANFIS>ANT-ANFIS> GA-ANFIS> ANFIS. From the ordering we can confirm that the proposed technique works well in software reliability prediction.

Table.1: *Comparison of reliability values with different techniques*

No of Component	Faults	Predicted Reliability					
		Proposed CSO - ANFIS	ABC -ANFIS	PSO -ANFIS	GA ANFIS	ANT - ANFIS	ANFIS
Component 1	5	0.88	0.83	0.82	0.77	0.78	0.70
Component 2	2.5	0.89	0.83	0.80	0.70	0.76	0.70
Component 3	4	0.86	0.85	0.84	0.75	0.76	0.62
Component 4	2	0.91	0.89	0.87	0.75	0.79	0.67
Component 5	1.6	0.97	0.90	0.91	0.80	0.78	0.75
Component 6	2.8	0.89	0.85	0.78	0.73	0.79	0.72
Component 7	3	0.89	0.85	0.80	0.70	0.83	0.704
Component 8	0.8	0.98	0.95	0.85	0.67	0.78	0.67
Component 9	3	0.94	0.88	0.89	0.74	0.72	0.68
Component 10	2	0.95	0.94	0.90	0.70	0.776	0.69

To evaluate the proposed technique, Root Mean Square Error (RMSE), Average Error (AE), Mean Absolute Error (MAE), Mean absolute percent error (MAPE) and Normalized Mean Square Error (NRMSE) are calculated. The graph is drawn by taking average values of ten numbers of components. The performance reliability prediction can be easily identified using RMSE and MSE. The greater variation between RMSE and MSE denotes that there is some error in the prediction techniques. If the RMSE and MSE values are equal, then it proves that all the measures are correct. With this concept, on looking at the graph the RMSE and MSE values are equal for the proposed technique, where as the ABC-ANFIS technique has slight variation between RMSE and MSE. This proves the efficiency of the proposed technique. While looking at the other error values, the proposed technique has lower error rates than that of the compared techniques.

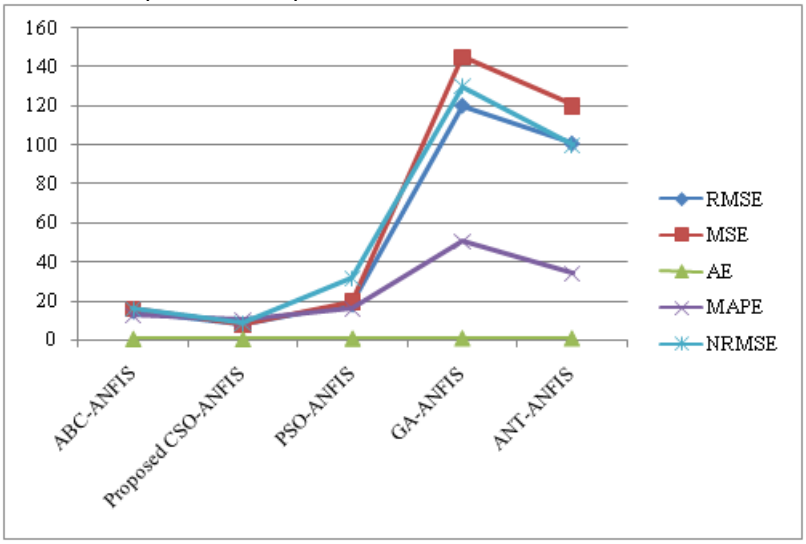

Fig.3*: Comparison of RMSE, MSE, AE, MAE, MAPE and NRMSE for various techniques*

4.1. Benchmark Functions

Although the results obtained for the engineering problems prove that CSA shows a competitive performance with the conventional algorithms, but still, there exists a question regarding the performance of CSA in larger-scale problems. In order to evaluate the performance of CSA on larger-scale optimization problems, three well-known benchmark functions shown in Table.2 are solved in 10 dimensions.

Sphere Function:

The Sphere function has d local minima except for the global one. It is continuous, convex and unimodal. The function is usually evaluated on the hypercube $x_i \in [-5.12, 5.12] \forall i = 1,..., d$.

$$f(x) = x_1^2 + x_2^2 + ..., + x_d^2 \quad (13)$$

Rosenbrock Function:

The Rosenbrock function, also referred to as the Valley or Banana function is a popular test problem for gradient-based optimization algorithms. It is shown in the plot above in its two-dimensional form. The function is unimodal, and the global minimum lies in a

narrow, parabolic valley. However, even though this valley is easy to find, convergence to the minimum is difficult. The function is usually evaluated on the hypercube $x_i \in [-5,10]\forall i = 1,..., d$, although it may be restricted to the hypercube $x_i \in [-2.048 ,2.048]\forall i = 1,..., d$.

$$f(x) = \sum_{i=0}^{d} \left[100 \left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right] \quad (13)$$

Table.2: *Evaluation and comparison of the CSA with the conventional algorithms using benchmark function*

Function	Index	CSA	PSO	GA	ABC	ANT
Sphere	Best	$9.58x10^{-13}$	$6.45x10^{-7}$	0.09	$7.88x10^{-10}$	$5.79x10^{-8}$
	Mean	$4x10^{-11}$	$3.10x10^{-5}$	2.01	$3.55x10^{-6}$	$2x10^{-3}$
	Std	$6.17x10^{-11}$	$4.54x10^{-5}$	2.21	$5.23x10^{-8}$	$1.28x10^{-3}$
	Avg time (sec)	0.67	0.98	1.86	1.23	1.05
RosenbrockFunction	Best	1.52	2.85	42.98	2.1	10.12
	Mean	10.86	18.33	496.78	13.43	77
	Std	22.76	39.43	769	30.14	300
	Avg time (sec)	0.87	1.11	1.91	1.11	1.8
Griewank Function	Best	0.0099	0.01	0.41	0.02	0.3
	Mean	0.21	0.12	0.86	0.43	0.99
	Std	0.12	0.08	0.20	0.13	0.19
	Avg time (sec)	1.14	1.37	2.15	1.22	1.34

Griewank Function:

The Griewank function has many widespread local minima, which are regularly distributed. The function is usually evaluated on the hypercube $x_i \in [-600 ,600]\forall i = 1,..., d$.

$$f(x) = \sum_{i=0}^{d} \left[\frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1\right] \quad (14)$$

The performance measures of the benchmark functions are tabulated in the below table 1. When compared with the conventional algorithms such as PSO, GA, ABC and ANT, the proposed technique provides better results. In sphere function, on looking at the results, GA provides lower performance and CSA provides better performance. When compared to ABC and CSA, the results of ABC look closer to CSA. Similarly in the other two benchmark functions namely Rosenbroke and Griewank, CSA provides better performance. Hence it proves that CSA works well and it can work with larger applications.

5. CONCLUSION

The software development process becomes increasingly time-consuming and expensive due to the complexity of software systems. In the mean time, the need for the highly reliable software system is ever increasing. How to enhance the reliability of the software systems and reduce the costto an acceptable level becomes the main focus of the software industry. In this paper, we have proposed a adaptive technique which combines a well known optimization technique named CSO and classification technique named ANFIS with low cost. The parametersof the ANFIS technique are optimized with the help of CSO in order to avoid the falling of local minima and increase the convergence rate which increases the predictability with less iteration. The performance of the proposed technique is evaluated using the banking sector data and it is compared with conventional techniques such as ABC-ANFIS, PSO -ANFIS, GA-ANFIS, ANT-ANFIS and ANFIS. Also the performance of the optimization technique is evaluated using the bench mark functions Sphere, Griewank and Rosenbrock respectively which shows the efficiency of the proposed optimization algorithm. Hence the

proposed technique can be used in real time software reliability prediction.

REFERENCES

[1] Srinivasan Ramani ,Swapna S. Gokhale , Kishor S. Trivedi, "SREPT: software reliability estimation and prediction tool", Performance Evaluation, Vol.39, pp. 37–60, 2000

[2] Mohammad Azzeh and Ali BouNassif, "Analyzing the Relationship between Project Productivity and Environment Factors in the Use Case Points Method", Vol. 29, No. 9, September 2017

[3] PanagiotisSentas, Lefteris Angelis, IoannisStamelos, and George Bleris, "Software productivity and effort prediction with ordinal regression", Information and Software Technology, Vol. 47, pp. 17–29, 2005

[4] Ali BouNassif, Luiz Fernando Capretz, and Danny Ho, "Estimating Software Effort Using an ANN Model Based on Use Case Points", In proceedings of 11th International IEE conference on Machine Learning and Applications, 2012

[5] ZaheedMahmood, DavidBowes, TracyHall, Peter C.R.Lane, JeanPetrić, "Reproducibility and replicability of software defect prediction studies", Information and Software Technology, Vol. 99, pp. 148-163, July 2018

[6] Martin Shepperdaand Steve MacDonel, "Evaluating prediction systems in software project estimation"", Information and Software Technology, Vol. 54, pp. 820–827

[7] Heejun Park and SeungBaek, "An empirical validation of a neural network model for software effort estimation", Expert Systems with Applications, Vol. 35, pp. 929–937, 2008

[8] RadekSilhavy, PetrSilhavy, ZdenkaProkopova, "Analysis and selection of a regression model for the Use Case Points method using a stepwise approach", Journal of Systems and Software, Vol. 125, pp. 1-14, March 2017

[9] Sun-Jen Huang, Nan-Hsing Chiu and Li-Wei Chen, "Integration of the grey relational analysis with genetic algorithm for software effort estimation", European Journal of Operational Research, Vol. 188, pp. 898–909, 2008

[10] J.M. Verner, W.M. Evancoand N. Cerp, "State of the practice: An exploratory analysis of schedule estimation and software project success prediction", Information and Software Technology, Vol. 49, pp. 181–193, 2007

[11] Yu-Shen Su and Chin-Yu Huang, "Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models", The Journal of Systems and Software, Vol. 80, pp. 606–615, 2007

[12] Petra Vizarreta, Kishor Trivedi, BjarneHelvik, PoulHeegaard, Andreas Blenk, Wolfgang Kellerer, and Carmen Mas Machuca, "Assessing the Maturity of SDN Controllers with Software Reliability Growth Models", IEEE Transactions on Network and Service Management, Vol. 15, No. 3 , Sept. 2018

[13] IngunnMyrtveit, Erik Stensrud, Member, IEEE, and Martin Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models", IEEE Transactions On Software Engineering, VOL. 31, NO. 5, MAY 2005

[14] C. Manjula, and Lilly Florence, "Deep neural network based hybrid approach for software defect prediction using software metrics", Cluster Computing, pp. 1-17, 2018

[15] T. Ravi Kumar, T. Srinivasa Rao and SandhyaBathini, "A Predictive Approach to Estimate Software Defects Density Using Weighted Artificial Neural Networks for the Given Software Metrics", Smart Intelligent Computing and Applications, pp.449-457, 2019

[16] ParthaSarathiBishnu and VandanaBhattacherjee, Software Fault Prediction Using Quad Tree-Based K-Means Clustering Algorithm", IEEE Transactions On Knowledge and Data Engineering, Vol. 24, No. 6, June 2012

[17] T aghi M. Khoshgoftaar and NaeemSeliya, "Tree-Based Software Quality Estimation Models For Fault Prediction"', In proceedings of proceedings Eighth IEEE Symposium on Software Metrics, 2002.

[18] Subhashis Chatterjee and BappaMaji, "A bayesian belief network based model for predicting software faults in early phase of software development process", Applied Intelligence, Vol. 48, No. 8, pp 2214–2228, August 2018

[19] KapilJuneja, "A fuzzy-filtered neuro-fuzzy framework for software fault prediction for inter-version and inter-project evaluation", Applied Soft Computing Journal, Vol. 77, pp. 696–713, 2019

[20] Cong Jin and, Shu-Wei Jin,"Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms", Applied Soft Computing, Vol. 15, pp. 113–120, 2014

[21] ArunimaJaiswaland Ruchika Malhotra, "Software reliability prediction using machine learning techniques"', International Journal of System AssurEngManag, Vol.9, No. 1, pp. 230-244, February 2018

[22] Subhashis Chatterjee, BappaMaji and Hoang Pham, "A fuzzy rule-based generation algorithm in interval type-2 fuzzy logic system for fault prediction in the early phase of software development", Journal of Experimental & Theoretical Artificial Intelligence, 2018

[23] Mengmeng Zhu and Hoang Pham, "A Two-Phase Software Reliability Modeling Involving with Software Fault Dependency and Imperfect Fault Removal", Computer Languages, Systems & Structures, Vol. 53, pp. 27-42, Sep 2018

[24] Chander Diwaker1 & Pradeep Tomar2 & Ramesh C. Poonia3 &Vijander Singh, "Prediction of Software Reliability using Bio Inspired Soft Computing Techniques", Journal of Medical Systems, pp. 42:93, 2018

[25] NeelamdhabPadhy, R.P.Singh and Suresh ChandraSatapathy, "Software reusability metrics estimation: Algorithms, models and optimization

techniques", Computers & Electrical Engineering, Vol. 69, pp. 653-668, July, 2018

[26] TirimulaRaoBenala and RajibMall, "DABE: Differential evolution in analogy-based software development effort estimation", Swarm and Evolutionary Computation, Vol. 38, pp. 158-172, Feb 2018

[27] AndrésGarcía-Floriano, CuauhtémocLópez-Martín, CornelioYáñez-Márquez, and AlainAbran, "Support vector regression for predicting software enhancement effort", Information and Software Technology, Vol. 97, pp. 99-109, May 2018

[28] Hamza Turabieh, MajdiMafarja and Xiaodong Li, "Iterated feature selection algorithms with layered recurrent neural network for software fault prediction", Expert Systems with Applications, Vol. 122, pp. 27–42, 2019