STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720 (Online)
Volume 5, Issue 1
www.stmjournals.com

A Novel Software Cost Estimation Model Based on the Hybrid of Artificial Neural Network and Firefly Algorithm

Zahid Hussain Wani[1,*], S.M.K. Quadri[2]
[1]Department of Computer Science, University of Kashmir, Srinagar, Jammu and Kashmir, India
[2]Department of Computer Science, Jamia Millia Islamia, Jamia Nagar, New Delhi, India

Abstract

Software cost estimation is the prediction of software development effort and software development time required to develop a software project. Software cost estimation is a basic and important task for both successful execution of software development life cycle and its management in terms of its cost and time. Accurate software cost estimation is considered to be a difficult task as the information about the software project to be developed at the time of its inception and conclusion remains vague, thus prompts the researchers from both academics and industry to explore in the same. In this paper, we present a novel hybrid model of multi-layer artificial neural network and firefly algorithm for the purpose of getting accurate estimation of software development costs. Artificial Neural Networks, because of their capabilities in self-learning, modeling complex nonlinear relationships, fastness and fault tolerance against noise, are considered to be a very powerful in giving solution to prediction problems. In our proposed study, we are using multilayer layer artificial neural network as our core architecture for developing the accurate estimates of software development costs and firefly algorithm as its training algorithm because of its multimodal optimization capability. The proposed model has been evaluated using two basic evaluating criteria namely magnitude of relative error and median of magnitude of relative error as a measure of performance index to simply weigh the obtained quality of estimation.

Keywords: *Artificial neural network, bat algorithm, cuckoo search, functional link artificial neural network, firefly optimization, genetic algorithm, software cost estimation*

****Author for Correspondence*** E-mail: zahid_scholar@kashmiruniversity.ac.in

INTRODUCTION

Software cost estimation is the summation of predictions of both building effort and calendar time used to develop a software project. The building effort includes the sum of working hours and the total number of workers included in the process of soft project development. Just from the inception of software project development, organizations of this nature came across the problem of poor estimations of development effort and development time of software projects. A good reason for this was and which is persistent even this time, is the availability of vague information about the software project to be developed at the time of its estimation process. A good software cost estimation model helps project managers to estimate exact delivery time and accurate development cost required for any software development project. Moreover, a better estimate is the only thing that can let any software development project manager to evaluate the project progress, gives a track of potential cost control and delivery accuracy. This in widespread can give the organization a better insight of resource utilization and consequently will land the organization in a better schedule of its futuristic projects. For this purpose, a good number of software cost estimation techniques from last so many decades have been proposed which differ from algorithmic to non algorithmic to data mining techniques and to metaheuristics algorithms, but unfortunately, none among these satisfy the acceptance standard of accuracy in estimation of software development projects [1–2]. However, a varying range of artificial neural network models has also been developed for this purpose which however has shown improvements in the said. A big advantage of proposed model is that software cost

estimation using artificial neural network utilizes a supervised learning called back propagation for the purpose of training the network whereas the multi-layered structure in a multilayer layer perceptron lets down its training speeds to a minimum [3]. Thus, problems of the nature of overfitting and interference of weights let the training of a multilayer layer perceptron network difficult, which in this study are removed by introducing the firefly algorithm [4]. The results obtained from the proposed technique have been evaluated using magnitude of relative error and median of magnitude of relative error and later compared with two existing techniques. The results obtained from the proposed techniques have shown a significant upright in terms of accuracy in software cost estimation.

The rest of the paper is organized as: Next part of the paper presents an overview of the existing literature, proposed artificial neural network model for software cost estimation followed by firefly optimization algorithm. The following part reflects upon the data sets and the evaluation criteria based on which the different existing techniques in-addition to the proposed technique will be assessed and evaluated, reports on the experimentation results and their analysis generated from the newly proposed technique, followed by conclusion and future work.

RELATED WORK

There are so many ambiguities in software development process. A big reason for this is the large scale of dependent factors on which software development is based. This scale of factors may include the availability of large number of programming environments in which a software product can be developed or it can be the experience/inexperience of the development team and rest others. So, estimating a software development cost is complex and challenging too but equally demands a good attention as the success of any software development project can only be sustained by its precise cost estimation process [5].

The very initial software cost estimation approaches were introduced in the late 1960s and were based on expert judgment [6]. Expert judgment is an amorphous process where in domain experts devise their decisions based on their knowledge, wisdom and experience for the reason of estimation of software cost. The domain experts here do not need any history data for this purpose [7].

From past three and half decades now, a good number of software cost estimation models have been introduced. These include the models like Boehm's Cocomo [1], Cocomo II [2], Putnam's SLIM [5], and Albrecht's Function Points Analysis [8] and some others, even based on software requirements specifications [9]. All these models provide a formulaic foundation for software effort estimation [10], and thus allow the results for their analysis and evaluation.

Among all of the above mentioned formal models, COCOCMO is believed to be the most consistent software cost estimation model of the time and even the highly cited one but did not succeed in meeting the development standards and environment of late 1990s. It was developed by Barry Boehm in 1981 and is a regression-based model. COCOMO considers 17 cost drivers [5]. These cost drivers are rated on a scale from very low to extra high. This model is given below:

$$\text{Effort} = \text{A} * [\text{SIZE}]^{\text{B}} * \prod_{i=1}^{17} \text{EM}_i$$

Where;

$$\text{B} = 1.01 + 0.01 * \sum_{j=1}^{5} (\text{Scale Factor})_j$$

In above equation, 'A' means multiplicative constant and 'SIZE' depicts the size of the software calculated as selection of scale factors (SF) and EM*i* is a set of 14 effort multipliers.

All of the above mentioned models require accurate guesses of attributes like line of code (LOC), number of user screens, interfaces, complexity etc. Attributes like these cannot be easily acquired at the very early stage of software development process. However, still these acquired models are unable to handle categorical data and also are deficient in their reasoning capabilities. A big reason for this is

the inherent complex relationships between these attributes. So, understanding these models is equally difficult as their calculations. Thus their limitations lead to the exploration of a range of machine learning techniques [11–13] and other data mining techniques [14].

These data intensive techniques include various regression techniques resulting in linear models [15], nonlinear models like neural networks [16], rule based models [17] and lazy learning approaches [18]. The outcomes of these data mining techniques are the formulas which are objective in nature and also allow for their analytic evaluation. Also, these techniques are not limited to a specific set of attributes, as is the case with formal models like Cocomo I and owning such a subset of strong capabilities thus lead these models to get widely accepted in other researches too. There are many other researchers, who presented their research studies based on neural networks approach for the purpose of estimating software development effort [19–22].

PROPOSED HYBRID MODEL OF ARTIFICIAL NEURAL NETWORK AND FIREFLY ALGORITHM

There are many architectures of neural networks architecture that have been previously developed for different applications. A neural network architectural configuration rests on its design and its setting of performance constraints. Various parameters used in an ANN include:

1) An input layer and number of nodes in it.
2) Number of hidden layers and number of nodes in each hidden layer.
3) Training algorithm parameters.
4) Weights associated with each communication link present between neurons.

In our proposed study, 20 numbers of nodes are taken as inputs which represent 15 effort multipliers and 5scale factors. In our proposed network, a bias '*b*' is initialized at 1.00 and learning rate at 0.1. Moreover, in our proposed network, identity activation function is used to calculate the desired output of the network. After getting the results as an output from multilayer layer artificial neural network are repeatedly passed to FF algorithm for the training purpose of the network and the output if found optimal becomes the solution as a desired criteria of estimation, otherwise the process of training is continued till the solution is found. The desired algorithm is given below:

Step 1 Initialize bias, *b*=1, learning rate, lrate=0.1 and Threshold theta '*Θ*'=4.

Step 2 Until condition is false, execute steps 3–8.

Step 3 Execute step 5–7 for each training pair.

Step 4 The input layer gets information and transmits it to concealed layer by executing identity activation function on the inputs from x=1 to 20.

Step 5 Calculation of response of each response per unit as follows:

5.1 Calculating effort multiplier hidden layer using hidden_em=*b*+input×weight_em for input(x)=1 to 15.

5.2 Calculating scale factor hidden layer using hidden_sf=b+input× (weight_sf+size) for input (x)=16 to 20.

Step 6 Calculate estimated output = hidden_em×weight_em + hidden_sf×weight_sf.

Where, weight_em = effort multiplierweight and weight_sf = scale factor weight.

Step 7 If weights are not updated. Go to step 10.

Step 8 If the difference is in acceptable limit, the output is considered, else weights are modified using firefly algorithm.

Step 9 Repeat step 5 and step 6.

Step 10 Calculate Magnitude of Relative Error, MRE=$((Eff_{Actual}- Eff_{estimated})/Eff_{Actual})$ ×100.

Step 11 Stop.

The pictorial description of the proposed model is given as (Figure 1).

Firefly Algorithm (FA), introduced by Xin-She is a multi-modal optimization algorithm. It has been inspired from the behavior of fireflies or lightning bugs [23]. FA has been found to give solutions to many problems and has the significant competence over other metaheuristic algorithms to reach a solution. There are three basic rules that FA works on.

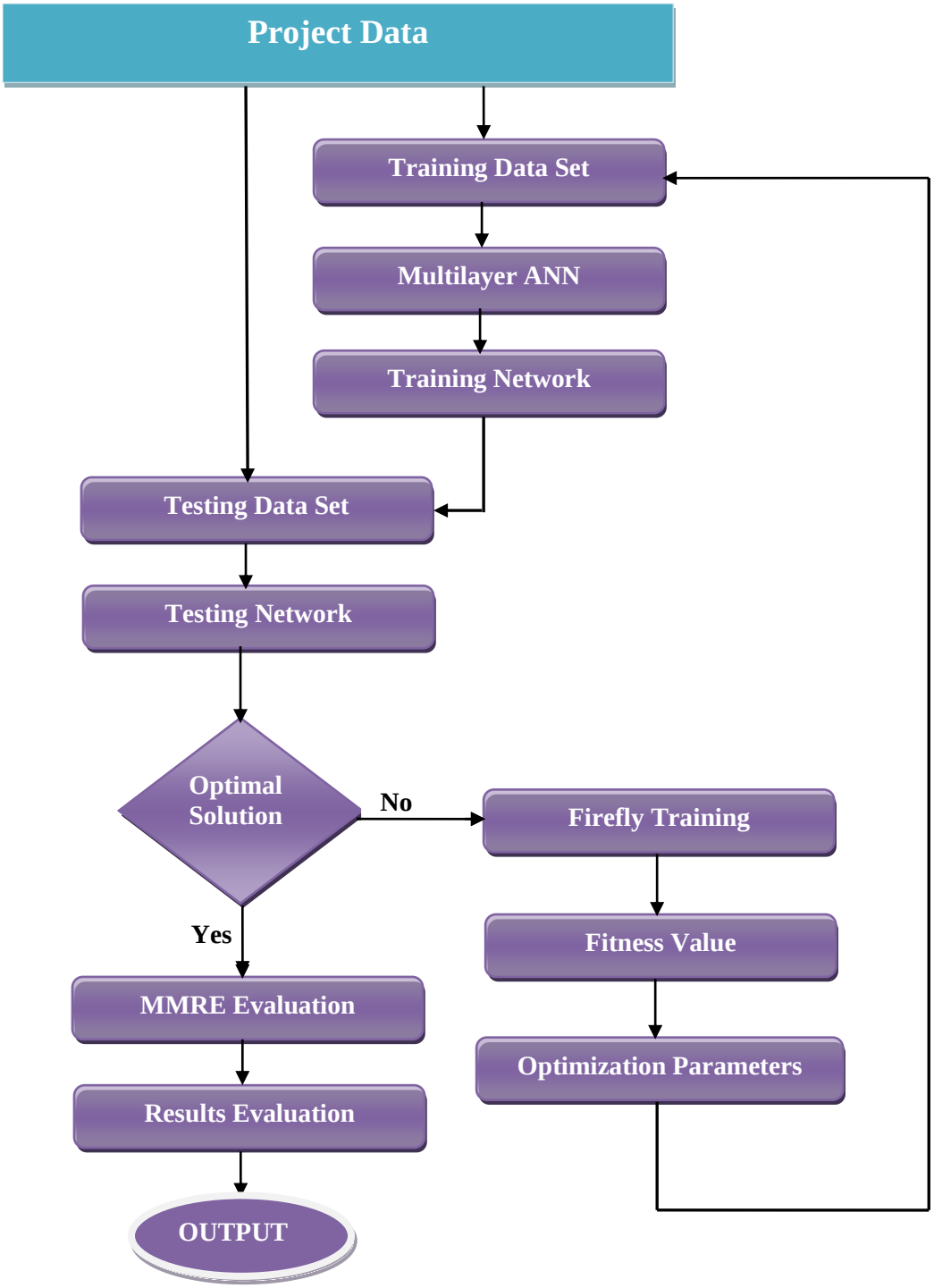

Fig. 1: *Flowchart of Proposed Model.*

The rules include:

Rule 01 This rule states that all fireflies are attracted to each other with irrespective of their gender.

Rule 02 The second rule states that attraction between more bright and less bright flies is correlated as bright flies attract less bright flies. This rule also states that in absence of brighter flies, the movement becomes random.

Rule 03 The last main rule states that the setting of the objective function determines or affects the light emission of the fly so that brightness is proportional to the objective function.

The pseudocode of the algorithm is given as:

Algorithm-Firefly Pseudocode

1. Objective function f(*x*);

2. x=(x1, x2,….., xd);
3. Generating Initial Population x_i; for all (i=1,……, n);
4. Light Intensity Ii at x_i is calculated by f (x_i);
5. Set Light absorption coefficient 'μ';
6. while (j<MaxGen)

do
for (k=1 to n all n fireflies)
do
for(l=1 to k all n fireflies)
do
if (I_l>I_k), then
Move firefly k towards l in d dimensions;
End if
Divergence in attractiveness with distance 'r' through exp[μr];
Evaluate new solutions and update light intensity;
end for
end for
Rank the fireflies and find the current best

7. end while
8. Post-process results and visualization

A big challenge in firefly algorithm is the modeling of attractiveness and modeling of intensities of light. For any firefly at location 'x', brightness 'I' is modeled as I (x) α f(x). While attractiveness 'μ' is proportional to the flies and is correlated to the distance $R_{i,j}$ between fireflies 'i' and 'j'. Below given equation demonstrates the inverse square of light intensity $I(r)$ wherein I_0 signifies the source light intensity.

$$I(r) = I_0 e^{-\mu r^2} \quad (1)$$

Presuming 'μ' as an absorption coefficient, the intensity is given as:

$$I(r) = \frac{I_0}{1+\mu r^2} \quad (2)$$

Where, 'I_0' is the original intensity.

The distance between a firefly at location x_k and another firefly at location x_l is calculated by the Euclidean distance given below in Eq. (3):

$$R_{kl} = \|x_k - x_l\| = \sqrt{\sum_{m=1}^{d}(x_{k.m} - x_{l,m})^2} \quad (3)$$

A firefly 'k' which gets attracted to firefly 'l' can be represented as:

$$x_k = x_k + \beta_{e^{\mu r_{kl}^2}}(x_l - x_k) + \alpha \left[rand - \frac{1}{2}\right] \quad (4)$$

Where, 'l' is brighter than 'k';
Moreover, the attractiveness variations are determined using 'β' which in-turn affects both behavior and convergence speed of firefly algorithm too.

Data Sets and Evaluation Criteria

Two data sets have been chosen in this study. One of the data sets has been got from the study of Mair *et al.* [24]. In this study, COCOMO 81 has been selected among a total set of 32 datasets as this dataset contains data of more than 50 software development projects. Another public domain data set Cocnasa was used in this study [25].

Accuracy in software cost estimation is evaluated based on the difference between the estimated effort and actual effort. A principle measure already used in the existing literature for these kinds of cost estimation models is magnitude of relative error [26].

$$MRE\ (in\ \%) = \frac{Actual\ Effort - Estimated\ Effort}{Actual\ Effort} * 100$$

MRE is calculated for each project. However, the average of MRE of 'n' projects using their mean is calculated as:

$$MMRE_{(Mean\ Magnitude\ of\ Relative\ Error)} = \frac{1}{n}\sum_{xi=1}^{n} MRE(xi)$$

Where; xi=1 to 'n' projects.
MMRE is the commonly used evaluation criteria however being much exposed to outliers [27], Median of Magnitude of Relative Error (MdMRE) is used as another evaluation criterion in this study instead of MRE. The MdMRE is given as:

$$MdMRE = 100 \times Median\ (MRE)$$

EXPERIMENTATION RESULTS

This section presents the experimentation results of the proposed model. The magnitude of relative error value of proposed model is calculated for randomly selected set of projects from all of the two datasets and then compared with the results obtained using

COCOMO-II model which is considered to be the basic estimation model in software development projects. Below two tables namely Tables 1 and 2 represent a significant substantial difference in MRE of the proposed technique against COCOMO-II model.

Accordingly, based on their MRE values, graphical demonstration as shown in Figures 2 and 3 are used to illustrate the performance of two existing and proposed model when executed on COCOMO 81 and Cocnasa datasets.

The MdMRE is calculated from the obtained results whilst applying both COCOMO model and the proposed model on both of the datasets. The MdMRE is given in the following Table 3 followed by its bar chart description as given in Figure 4.

Table 1: *MRE (%) of Proposed Model and other Two Existing Techniques on 11 Randomly Selected Projects of COCOMO81 Dataset.*

S No.	Project Id	MRE (%) on COCOMO81 Dataset	
		COCOMO-IIModel	Proposed Model
01	05	7.44	3.51
02	12	19.83	6.12
03	30	6.49	3.10
04	38	50.98	14.10
05	40	12.40	4.19
06	45	5.35	2.62
07	47	16.40	3.01
08	59	8.66	2.66
09	61	13.10	4.02
10	62	6.22	2.70

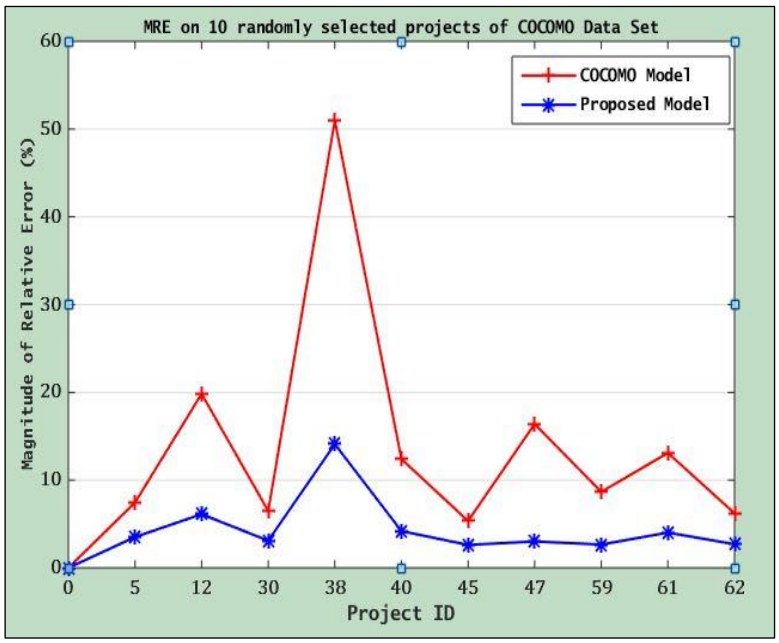

Fig. 2: *Graphical Description of MRE (%) of COCOMO and Proposed Model on COCOMO Data Set.*

Table 2: *MRE of Proposed Model and other Two Existing Models on 10 Randomly Selected Projects of Cocnasa Dataset.*

S No.	Project Id	MRE (%) on Cocnasa Dataset	
		COCOMO-IIModel	Proposed Model
01	1	9.33	2.98
02	5	8.84	2.19
03	15	16.75	5.44
04	25	14.09	3.58
05	30	8.81	3.10
06	42	13.9	4.10
07	54	13.67	4.50
08	60	11.78	5.10
09	62	13.2	2.10
10	75	17.09	4.50

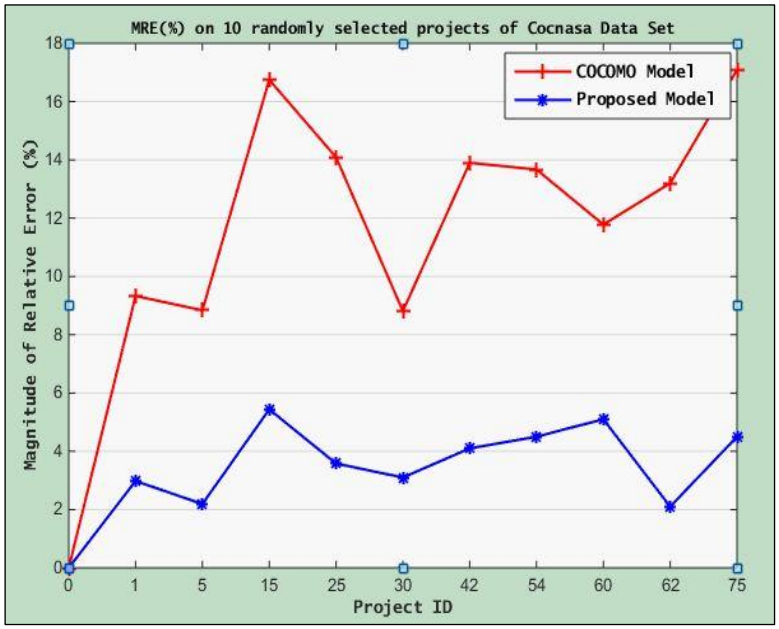

Fig. 3:*Graphical Description of MRE (%) of COCOMO and Proposed Model on Cocnasa Data Set.*

Table 3: *MdMRE of Proposed Model and COCOMO Model on 10 Randomly Selected Projects of COCOMO and Cocnasa Datasets.*

	MdMRE Comparison	
	COCOMO-II Model	**Proposed Model**
COCOMO Dataset	10.53	3.30
Cocnasa Dataset	13.43	3.84

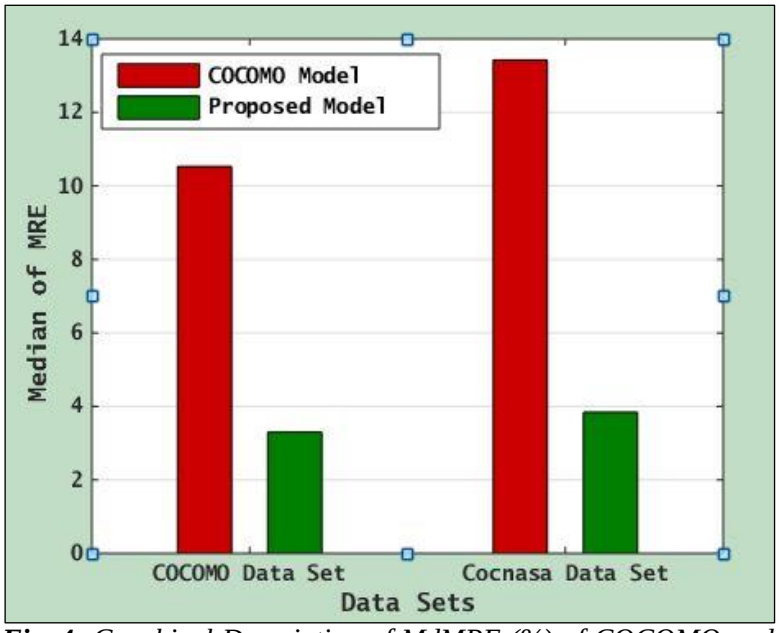

Fig. 4: *Graphical Description of MdMRE (%) of COCOMO and Proposed Model on both COCOMO and Cocnasa Data Sets.*

CONCLUSION AND FUTURE WORK

This research study concludes that estimating software development cost by introducing above proposed hybrid model of artificial neural network and firefly algorithm produces accurate results as the inclusion of firefly algorithm results in a model which intensifies the exploitations, thus makes a good possible balance of exploitation-exploration process which otherwise is not possible while using artificial neural networks only. Additionally, the proposed hybrid model improves the convergence rate to optimal region too. Moreover, the proposed model, in addition to attainment of accurate software cost estimation results, also trims the computational complexity without any forfeit on the performance.

In future however, it is possible to have some other hybridization of different architectures of artificial neural networks with other metaheuristic algorithms like artificial bee colony algorithm, ant colony optimization, cuckoo search, bat search and many alike to get even more positive results up in the domain of software cost estimation.

REFERENCES

1. Boehm BW. *Software Engineering Economics*. Englewood Cliffs, New Jersey: Prentice-Hall; 1981.
2. Boehm BW. *Software Cost Estimation with COCOMO II*. Englewood Cliffs, New Jersey: Prentice Hall PTR; 2000.
3. Patra JC, Pal RN. A Functional Link Artificial Neural Network for Adaptive Channel Equalization. *Sig Process*. 1995; 43: 181–195p.
4. Dehuri S, Cho SB. A Comprehensive Survey on Functional Link Neural Networks & an Adaptive PSO-BP Learning for CFLNN. *Neural Comput Appl*. 2010; 187–205p.
5. Putnam LH. A General Empirical Solution to the Macro Software Sizing and Estimating Problem. *IEEE Trans Soft Eng.* 1978; 4(4): 345–361p.
6. Nelson EA. *Management Handbook for the Estimation of Computer Programming Costs.* System Developer Corp.; 1966.
7. Booker JM, Meyer MM. Elicitation and Analysis of Expert Judgment. Los Alamos National Laboratory, LA-UR-99-1659.
8. Albrecht AJ, Gaffney JE. Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Trans Softw Eng*. Nov 1983; 9(6): 639–648p.
9. Ashish Sharma Vardhan, Dharmender Singh Kushwaha. A Versatile Approach for the Estimation of Software Development Effort Based on SRS Document. *Int J Softw Eng Knowl Eng*. 2014; 24(01): 1–42p.

10. Kemerer CF. An Empirical Validation of Software Cost Estimation Models. *Commun ACM*. 1987; 30(5): 416–429p.
11. Tirimula Rao B, Sameet B, Kiran Swathi G, *et al.* A Novel Neural Network Approach for Software Cost Estimation using Functional Link Artificial Neural Network (FLANN). *International Journal of Computer Science and Network Society (IJCSNS)*. 2009; 9(6): 126–131p.
12. Reddy CS, Raju KVSN. An Improved Fuzzy Approach for COCOMO's Effort Estimation using Gaussian Membership Function. *J Software (JSW).* 2009; 4(5): 452–459p.
13. Gharehchopogh FS. Neural Networks Application in Software Cost Estimation: A Case Study. *2011 International Symposium on Innovations in Intelligent Systems and Applications (INISTA 2011)*, IEEE, Istanbul, Turkey. 15–18 Jun 2011; 69–73;.
14. Jørgensen M, Shepperd M. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Trans Software Eng*. Jan 2007; 33(1): 33–53p.
15. Sentas P, Angelis L, Stamelos I, *et al.* Software Productivity and Effort Prediction with Ordinal Regression. *Inf Softw Technol*. 2005; 47: 17–29p.
16. G Finnie, Wittig G, Desharnais J-M. A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-Based Reasoning and Regression Models. *J Syst Softw*. 1997; 39: 281–289p.
17. Briand L, Emam KE, Surmann D, *et al.* An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. *Proc 21st Int'l Conf Software Eng*. May 1999; 313–323p.
18. Briand L, Langley T, Wieczorek I. A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. *Proc 22nd Int'l Conf Software Eng*. Jun 2000; 377–386p.
19. Hughes RT. An Evaluation of Machine Learning Techniques for Software Effort Estimation. University of Brighton; 1996.
20. Samson B, Ellison D, Dugard P. Software Cost Estimation Using an Albus Perceptron (CMAC). *Inf Softw Technol.* 1997; 39(1): 55–60p.
21. Heiat A. Comparison of Artificial Neural Network and Regression Models for Estimating Software Development Effort. *Inf Softw Technol*. 2002; 44(15): 911–922p.
22. Wittig G, Finnie G. Estimating Software Development Effort with Connectionist Models. *Inf Softw Technol*. 1997; 39(7): 469–476p.
23. Yang X-S. Firefly Algorithms for Multimodal Optimization. In: Watanabe O, Zeugmann T, editors. *Stochastic Algorithms*: *Foundations and Applications*, Vol. 5792 of *Lecture Notes in Computer Science*. Berlin and Heidelberg: Springer; 2009; 169–178p.
24. Shepperd Mair M, Jorgensen M. An Analysis of Datasets Used to Train and Validate Cost Prediction Systems. *ACM SIGSOFT Softw Eng Notes*. 2005; 4: 1–6p.
25. Menzies T, Chen Z, Hihn J, *et al.* Selecting Best Practices for Effort Estimation. *IEEE Trans Software Eng*. Nov 2006; 32(11): 883–895p.
26. Conte SD, Dunsmore HE, Shen VY. *Software Engineering Metrics and Models.* The Benjamin/Cummings Publishing Company, Inc.; 1986.
27. Port, Korte M. Comparative Studies of the Model Evaluation Criterions MMRE and PRED in Software Cost Estimation Research. *Proc Second ACM-IEEE Int'l Symp Empirical Software Eng and Measurement*. Oct 2008; 51–60p.

Cite this Article
Zahid Hussain Wani, Quadri SMK. A Novel Software Cost Estimation Model Based on the Hybrid of Artificial Neural Network and Firefly Algorithm. *Journal of Artificial Intelligence Research & Advances*. 2018; 5(1): 38–45p.