A Hybrid of Harmony Search and Functional Link Artificial Neural Network Model for Software Cost Estimation

Shazia Manzoor[1], **Zahid Hussain Wani** [2]

[1]Department of Computer Sciences,
Islamic University of Science & Technology, J&K, INDIA.
shazya.manzoor@gmail.com

[2] Department of Computer Sciences,
Islamic University of Science & Technology, J&K, INDIA.
zahid.uok@gmail.com

Abstract. Software cost estimation predicts the amount of effort and development time required to build any software system. There are a number of cost estimation models. Each of these models have their own pros and cons in estimating the development cost and effort. Recently, the usage of Meta-heuristic techniques for software cost estimations is increasingly growing. Prerequisite of relative accurate estimation is work experience. Therefore, the risk associated with construction of software projects is based on preliminary estimates. Increasing risk causes increasing uncertainty about the initial program with increasing complexity and size of projects level of uncertainty become higher. So in this paper, we propose an approach, which consists of Functional Link ANN and Harmony search algorithm as its training algorithm. FLANN reduces the computational complexity in multilayer neural network. It does not have any hidden layer, and has got the fast learning ability. In our work, MRE, MMRE and MdMRE were selected as the performance index to gauge the quality of prediction of the proposed model. Extensive evaluation of results show that training a FLANN with harmony search algorithm for the Software Cost prediction problem gives out highly improved set of results. In addition the proposed model is structurally simple and requires less computation during training as the model contains no hidden layer.

Keywords: Artificial Neural Network (ANN), Harmony Search, Functional Link Artificial Neural Network(FLANN), Software Cost Estimation(SCE).

Introduction

Software cost estimation is the prediction of amount of effort and time required to develop any software project. Software Development Effort simply covers the hours of work and the number of workers interms of staffing levels needed to develop a software project. Every single project manager is always in a quest of finding better estimates of software cost so that he can evaluate the project progress, has potential cost control, delivery accuracy, and in addition can gave the organization a better insight of resource utilization and consequently will land the organization in a better schedule of futuristic projects. The software effort prediction facet of software development project life cycle is always made at very early stage of software development life cycle. However, estimating the software development project at this time is most difficult to obtain because in addition to availability of sparse information about the project and the product at this stage, the so presented information is likely to be vague for the estimation purpose. So, estimating software development effort remains an intricate problem and thus continues to attract many researchers interest. A range of cost estimation techniques have been proposed so far and have been classified into various broad categories which will be discussed in the later sections. So far variations of a range of Artificial Neural Network models have also been developed for this purpose in addition to previously used Algorithmic models.
In this paper, our focus will be on a specially varied ANN called as Functional Link Artificial Neural Network (FLANN) which is infact a high order Artificial Neural Network model and in-particular its training algorithm which will be Harmony search algorithm for the purpose of achieving a better software cost estimation model.
The rest of the paper is organised as follows. In Section 2, the evolution of software cost estimation using both algorithmic and non-algorithmic models will be discussed. In addition characteristics of COCOMO II model will be presented here. FLANN architecture and the training algorithm implemented will be presented in sections 3. In Section 4 Datasets and Evaluation criteria will be discussed. Section 5 reports the Experimentation Results and their Analysis. Finally, Section 6 concludes the paper along with Future Work.

Software Cost Estimation Models

Software cost estimation is an uncertain task, as it is based on large set of factors which are normally vague in nature and is thus hard to model them mathematically. Software schedule and cost estimation supports the planning and tracking of software projects [1]. The very steadfast and precise software cost estimation process of any software development project in software engineering is always an ongoing challenge because of the reason that it allows for any organization's substantial financial and tactical planning. Software cost estimation techniques are classified into various categories and they include:
2.1 Parametric models or Algorithmic models are the models which are based on the statistical analysis of historical data. These models use a mathematical formula to predict project cost based on the estimates of project size, the number of software engineers, and other process and product factors [2]. These models can be built by analysing the costs and attributes of completed projects and finding the closest fit formula to actual experience. Some of the famous algorithmic models are: Boehm's COCOMO'81, II [2], Albrecht's Function Point [3] and Putnam's SLIM [1]. All of them require inputs, accurate estimate of specific attributes, such as line of code (LOC), number of user screen, interfaces, complexity, etc which are not

easy to acquire during the early stage of software development. Models based on historical data have limitations. Understanding and calculation of these models is difficult due to inherent complex relationships between the related attributes, are unable to handle categorical data as well as lack of reasoning capabilities. Thus their limitations lead to the exploration of the non-algorithmic techniques which are soft computing based. Among algorithmic models, COCOMO (Constructive Cost Model) is the most commonly used Algorithmic cost modelling technique because of its simplicity for estimating the effort in person-months for a project at different stages. COCOMO uses mathematical formulae to predict project cost estimation [1]. COCOMO is taken as a base model for the software cost estimation. Because of the importance of COCOMO Model in our research we provide a brief overview on this in the following sections.
The COCOMO model is a regression based software cost estimation model. It was developed by Barry Boehm [2] in 1981 and thought to be the most cited best known and the most plausible of all traditional software cost prediction models. COCOMO model can be used to calculate the amount of effort and the time schedule for software projects. COCOMO 81 was a stable model on that time. One of the problems with using COCOMO 81 is that it does not match the development environment of the late 1990's. Therefore, in 1997 COCOMO II was published and was supposed to solve most of those problems. COCOMO II has three models also, but they are different from those of COCOMO 81. They are
1) Application Composition Model – Suitable for projects built with modern GUI-builder tools. Based on new Object Points.
2) Early Design Model – To get rough estimates of a project's cost and duration before have determined its entire architecture. It uses a small set of new Cost Drivers, and new estimating equations. Based on Unadjusted Function Points or KSLOC.
3) Post-Architecture Model – The most detailed of the three, used after the overall architecture for the project has been designed. One could use function points or LOC as size estimates with this model. It involves the actual development and maintenance of a software product.
COCOMO II describes 17 cost drivers that are used in the Post-Architecture model [1]. The cost drivers for COCOMO II are rated on a scale from Very Low to Extra High in the same way as in COCOMO 81. COCOMO II post architecture model is given as:

$$\boldsymbol{Effort = A \times [SIZE]^B \times \prod_{i=1}^{17} Effort\ Multiplier_i} \qquad (1)$$

$$where\ B = 1.01 + 0.01 \times \sum_{j=1}^{5} Scale\ Factor_j$$

In equation 1, A: Multiplicative Constant, SIZE: Size of the software project measured in terms of the selection of scale factors (SF), is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation.
2.2 Expert judgment is a non-structured process in which experts make decisions to the technical problems based on their knowledge and experience in order to give accurate results than other techniques and doesn't require any previous historical data [4, 5, 6].
2.3 In 1990's non-algorithmic models were born and have been proposed to software cost estimation. Non Parametric models or Non-algorithmic models are the models which are based on fuzzy logic (FL), artificial neural networks (ANN) and evolutionary computation (EC). These models are used to group together a set of techniques that represent some of the facets of human mind [3], for example regression trees, rule induction, fuzzy systems, genetic algorithms, artificial neural networks, Bayesian networks and evolutionary computation. Tools based on ANN have increasingly gained popularity due to their inherent capabilities to approximate any nonlinear function to a high degree of accuracy. Neural networks are able to generalise from trained data set. A set of training data, a specific learning algorithm makes a set of rules that fit the data, and fits previously unseen data in a rational manner [7]. A lot of research has gone into the development of models based on a range of intelligent soft computing techniques over the last two decades. Early models employed the multi-layer perceptron (MLP) architecture using back propagation (BP) algorithm, while a lot of recent work is based on evolutionary optimization techniques such as genetic algorithms (GA).

Proposed FLANN model and its training using Harmony Search

FLANN model for predicting software development effort is a single-layer feedforward artificial neural network consisting of one input layer and one output layer. The FLANN generates output (effort) by expanding the initial inputs (cost drivers) and then processing to the final output layer. Each input neuron corresponds to a component of an input vector. The output layer consists of one output neuron that computes the software development effort as a linear weighted sum of the outputs of the input Layer. The reason of using FLANN is that multilayer feedforward network (MLP) though commonly used ANN in software cost estimation model utilizes a supervised learning technique called Backpropagation for training the network. However, due to its multi-layered structure, the training speeds are typically much slower as compared to other single layer feedforward networks [3]. Problems such as local minima trapping, overfitting and weight interference also make the network training in MLP challenging [8]. Hence, Pao [9] has introduced Functional Link Neural Network (FLANN) in avoiding these problems. This approach removes the hidden layer from the ANN architecture to help in reducing the neural architectural complexity and provides them with an enhancement in representation of input nodes for the network to be able to perform a better accuracy of software cost estimations. This network can be conveniently used for Functional approximation with faster convergence rate and lesser computational load than a multi-layer perceptron (MLP) structure. Because of its flatness and simplicity, the computations are simple. The functional expansion of the input to the network effectively increases the dimensionality of the input vector and hence the hyper-planes generated by the FLANN provide greater discrimination capability in the input pattern space [10]. A number of research papers on system identification and control of nonlinear systems, noise cancellation and channel equalization has been reported in recent times [11]. These experiments have proven the ability of FLANN to give out satisfactory results to problems with highly non-linear and dynamic data.

3.1 Architecture of FLANN: The FLANN network can be used not only for functional approximation but also for decreasing the computational complexity. This method is mainly focused on functional approximation. In the aspect of learning, the FLANN network is much faster than other network. The use of FLANN to estimate software development effort requires the determination of its architecture parameters according to the characteristics of COCOMO. A general structure of FLANN is shown in figure 1 followed by the detailed FLANN architecture in figure 2.

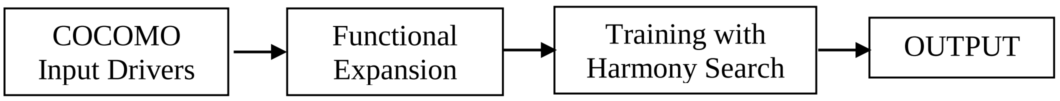

Figure 1-General structure of FLANN

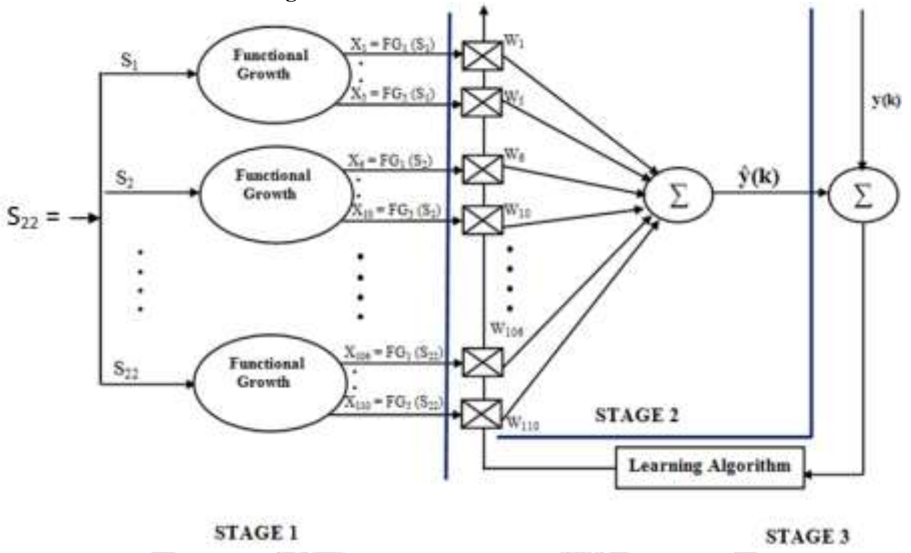

Figure 2: The Detailed FLANN Architecture

A detailed structure of FLANN is shown in figure 2. FLANN is a single-layer nonlinear neural network. Let k be the number of input-output pattern pairs to be learned by the FLANN. Let the input pattern vector X_k be of dimension n, and the output Y_k be a scalar. The training patterns are denoted by $\{X_k\ Y_k\}$. A set of N basis functions $Ø(X_k)=[Ø(X_k)\ Ø(X_k) \cdots Ø(X_k)\ ]^T$ are adopted to expand functionally the input signal $X_k=[x_1(k)\ x_2(k) \cdots x_n(k)\ ]^T$. These N linearly independent functions map the n-dimensional space into an N-dimensional space, that is $R^n \rightarrow R^N$, $n < N$.

The linear combination of these function values can be presented in its matrix form that is $S = WØ$. Here $S_k = [S_1(k)\ S_2(k) \cdots S_m(k)]^T$, W is the $m \times N$ dimensional weight matrix. The matrix S_k is input into a set of nonlinear function $\boldsymbol{\rho(\bullet) = tanh(\bullet)}$ to generate the equalized output $\hat{Y}= [\hat{y}_1\ \hat{y}_2 \cdots \hat{y}_m]^T$, $\hat{y}_j= \rho(S_j)$, $j =1, 2, \cdots, m$. The various steps involved here are:

Step-1: The 22 cost factors of the validated dataset are taken as the input of the network. These cost factors are then functionally expanded using the following formulas.

There are three different functional expansion of the input pattern in the FLANN. They are Chebyshev, Legendre and Power Series, corresponding networks named as C-FLANN, L-FLANN and P-FLANN respectively.

1) Chebyshev polynomials are given by:

$T_0(x) = 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1,\ T_3(x) = 4x^3 - 3x, T_4(x) = 8x^3 - 8x^2 + 1.$

Higher order Chebyshev polynomials may be generated by the recursive formula given as: $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x),\ n \geq 2,\ (-1 \leq x \leq 1).$

2) Legendre polynomials are given by:

$L_0(x) = 1,\ L_1(x) = x,\ L_2(x) = (3x^2-1)/2,\ L_3(x) = (5x^3 - 3x)/2,$

$L_4(x) = (35x^4 - 30x+3)/8.$

Higher order Legendre polynomials may be generated by the recursive formula given as: $L_n(x) = [(2n-1) x\ L_{n-1}(x)-(n-1)\ L_{n-2}(x)]/n,\ n\geq2,\ (-1\leq x\leq 1).$

3) Power series (x =any value)

$P_n(x) = x^n \quad , n \geq 0.$

Step 2: Let each element of the input pattern before expansion be represented as z(i) , $1 < i < S$ where each element z(i) is functionally expanded as $z_n(i)$,$1 < n < N$, where N = number of expanded points for each input element. In this study, N = 5 and S = total number of features in the dataset taken. Expansion of each input pattern is done as follows:

$x_0(z(i)) = 1,\ x_1(z(i)) = z(i),\ x_2(z(i)) = 2z(i)^2 - 1,$

$x_3(z(i)) = 4z(i)^3 - 3z(i),\ x_4(z(i)) = 8z(i)^3 - 8z(i)^2 + 1.$

Where, z(i), $1 < i < d$, d is the set of features in the dataset.

These nonlinear outputs are multiplied by a set of random initialized weights from the range [-0.5, 0.5] and then summed to produce the estimated output y(k).All the y(k)'s are summed to get ŷ(k).

Training with Harmony Search algorithm:

Inspired by the robustness and flexibility offered by the population-based optimization algorithm, we proposed the implementation of the Harmony search algorithm as the learning scheme to overcome the disadvantages caused by Backpropagation in the FLANN training. In order to create a new vector of variables (the harmony vector) in harmony search algorithm, each variable's value is determined according to one of the following rules [12]:

a) Select one of the existing values from the harmony memory
b) Select a value next to one of the values of harmony memory
c) Select the value for the variable from their legal constraint (out of harmony memory) randomly

Harmony search algorithm consists of five stages.

Stage 1- Defining the optimization problem and initializing algorithm parameters:

The optimization problem is maximizing or minimizing the objective function F(x); as a problem's variables stay within the legal constraints. Algorithm parameters which are initialized through this stage are: the number of decision variables (N) and it's change interval, harmony memory size (HMS), the probability of selecting a variable from harmony memory (HMCR), the probability of modifying the variable selected from harmony memory (PAR), modification bandwidth (BW) and algorithm termination criterion (largest frequency).

Stage 2- Initializing harmony memory:

Initializing the memory is completely random and the number of generated vectors matches the size of harmony memory. The primary harmony memory is saved with the values corresponding to the objective function.

Stage 3- Generating a new harmony from harmony memory:

The New Harmony vector is generated according to rules A, B and C.

Stage 4- Updating harmony memory:

If the new harmony is better than the worst existing harmony, it replaces the old one with the corresponding objective function value and the worst harmony of the memory is removed.

Stage5- Iteration of stage 3 and 4 until termination criterion is met:

The termination criterion is generally defined as satisfaction of one condition or reaching the highest frequency. This algorithm presents a new method for developing the new solution vector whose accuracy and convergence is improved by searching for harmony in increasing diversity in simple PAR. PAR parameter in increasing diversity in the beginning of the algorithm search within the solution space and the role of BW parameter in the local search of the algorithm play an important role in order to increase convergence. Among the disadvantages of the harmony search algorithm is the use of fixed values for parameters PAR and BW. In the improved harmony search algorithm, PAR and BW values change dynamically in each repeat.

The flowchart of the training algorithm is presented below in figure 3. In the initial process, the FLANN architecture (weight and bias) is transformed into objective function along with the training dataset. This objective function will then be fed to the harmony search algorithm in order to search for the optimal weight parameters. The weight changes are then tuned by the harmony search algorithm based on the error calculation (difference between actual and expected results).

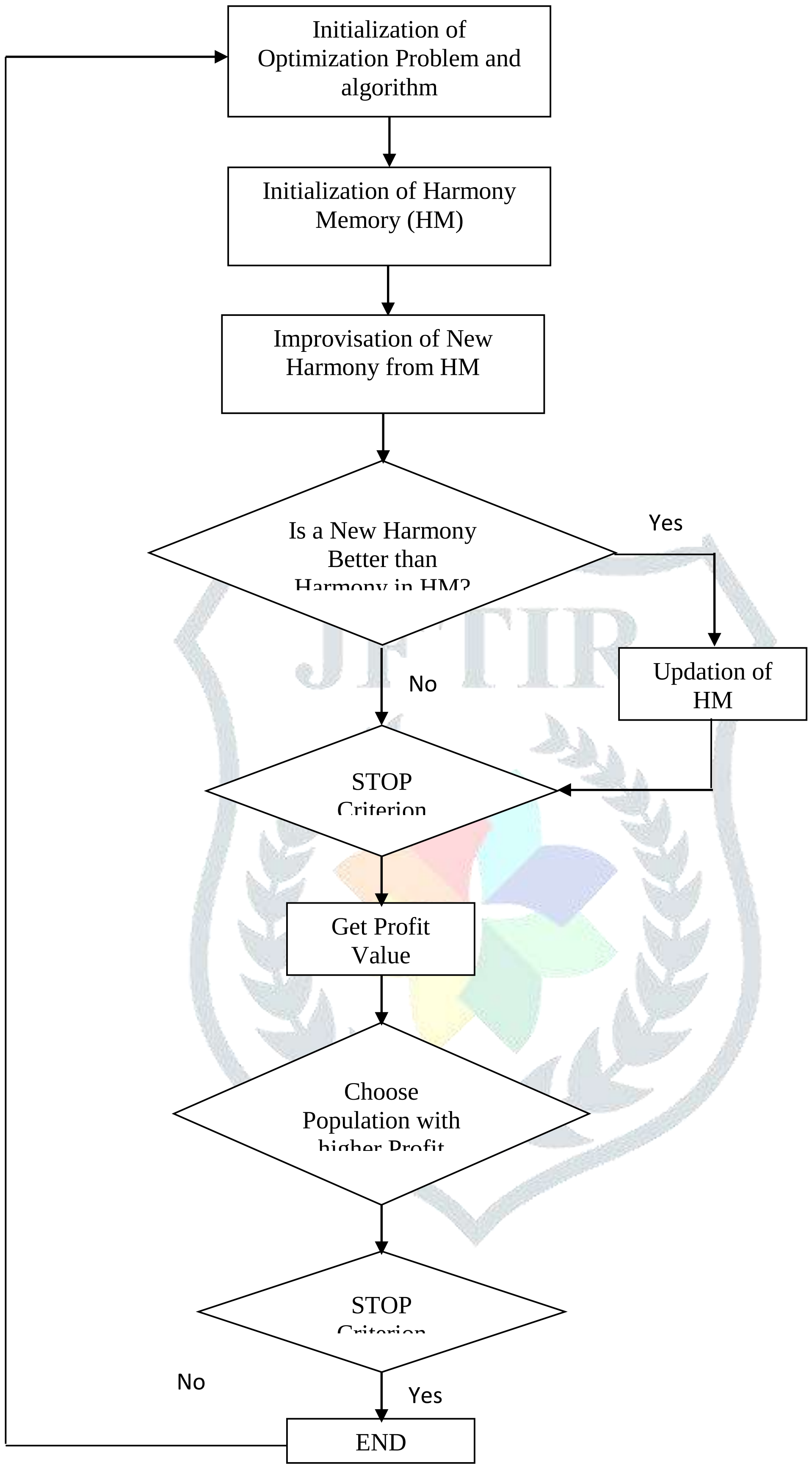
Initialization of Optimization Problem and algorithm
Initialization of Harmony Memory (HM)
Improvisation of New Harmony from HM
Is a New Harmony Better than Harmony in HM?
Yes
Updation of HM
No
STOP Criterion
Get Profit Value
Choose Population with higher Profit
STOP Criterion
No
Yes
END

Figure 3-General Harmony Search Flowchart

Datasets and Evaluation Criteria

The data sets used in the present study comes from PROMISE Software Engineering Repository data set [13] made publicly available for research purpose. The two datasets used are NASA 93 dataset and COCOMO_SDR. The NASA 93 dataset consists of 93 NASA projects from different centres for various years. It consists of 26 attributes: 17 standard COCOMO-II cost drivers and 5 scale factors in the range Very_Low to Extra_High, lines of code measure (KLOC), the actual effort in person months, total defects and the development time in months. The COCOMO_SDR dataset is from Turkish Software Industry. It consists of data from 12 projects and 5 different software companies in various domains. It has 24 attributes: 22 attributes from COCOMO II model, one being KLOC and the last being actual effort in person months. The entire dataset is divided into two sets, training set and validation set in the ratio of 80:20 to get more accuracy of prediction. The proposed model is trained with the training data and tested with the test data.

The evaluation consists in comparing the accuracy of the estimated effort with the actual effort. A common criterion for the evaluation of cost estimation model is the Magnitude of Relative Error (MRE) and is defined as in below Equation

$$MRE = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort}$$

The *MRE* values are calculated for each project in the validation set, while mean magnitude of relative error (MMRE) computes the average of MRE over *n* no. of projects.

$$MMRE = \frac{1}{n}\sum_{x=1}^{n} MRE$$

Since MRE is the most common evaluation criteria, and is thus adopted as the performance evaluator in the present paper. In addition the Mean Magnitude of Relative Error (MMRE) and the Median of MRE (MdMRE) for entire validation set is also calculated and compared with COCMO model in the present paper.

Experimentation Results and their Analysis

This section presents and discusses the results obtained when applying the proposed neural network model to the NASA 93 and COCOMO_SDR datasets. The model is implemented in Matlab. The MRE value is calculated for the projects in the validation set for all the three datasets and is then compared with the COCOMO model.

Below table 1 shows the results and comparison on NASA 93 dataset. There is a decrease in the relative error using the proposed model. For example, the relative error calculated for Project ID 30 is 8.81 for COCOMO model, and 3.34 for our proposed model. The relative error calculated for Project ID 62 is 13.2 for COCOMO model, and 5.00 for our proposed model. The Mean Magnitude of Relative Error (MMRE) for the entire validation set is 12.746 and 4.349 for the COCOMO model and our proposed model respectively. The MdMRE for the entire validation set is 13.43% for the COCOMO model and 4.46% for our proposed model.

S No	Project ID	MRE (%) using COCOMO Model	MRE (%) using proposed Harmony search based FLANN Model
1	1	9.33	3.80
2	5	8.84	3.29
3	15	16.75	4.45
4	25	14.09	3.77
5	30	8.81	3.22
6	42	13.9	5.13
7	54	13.67	4.49
8	60	11.78	5.21
9	62	13.2	4.70
10	75	17.09	3.88

Table 1-Comparison of MRE on NASA 93

Fig. 4 shows the graphical representation of MRE values for the two models for NASA 93 dataset. There is again a decrement in the relative error using the proposed model.

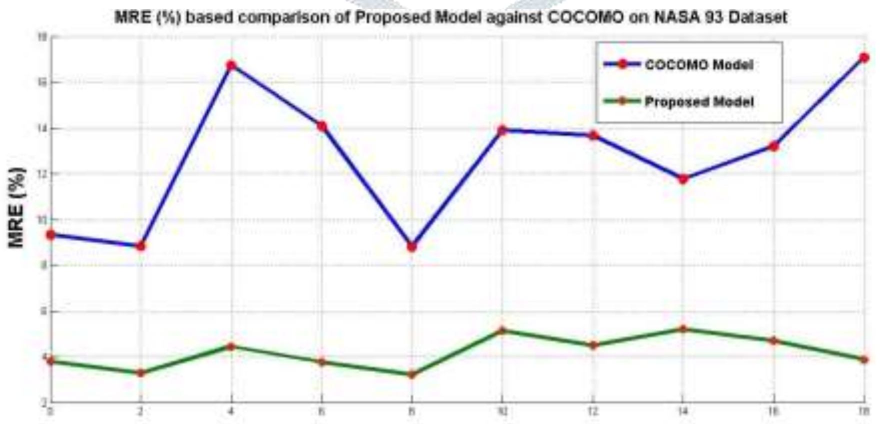

Figure 4 - Comparison of the Models on NASA93 dataset

Table 2 shows the estimated effort and their MRE values using the proposed model on COCOMO_SDR dataset. MMRE value for the estimated effort is 6.34. The MdMRE for the entire validation set is 4.62% for the proposed model.

S No	Project Id	Actual Effort	Estimated Effort	MRE (%)
1	1	1	1.22	0.22

2	2	2	1.67	0.16
3	3	4.5	4.29	0.046
4	4	3	2.94	0.02
5	5	4	3.67	0.082
6	6	22	20.12	0.085
7	7	2	1.93	0.035
8	8	5	3.67	0.266
9	9	18	16.73	0.070
10	10	4	3.84	0.04
11	11	1	1.06	0.06
12	12	2.1	2.12	0.009

Table 2: Comparison of effort on COCOMO_SDR

Figure 5 shows the bar graph representation of actual effort values and estimated effort values with the proposed model for COCOMO_SDR. The bar graph shows that the estimated effort is very close to the actual effort. The results obtained thus, suggest that the proposed model outperformed the COCOMO model in terms of all the discussed evaluation criteria i.e., MRE, MMRE and MdMRE. It thus can be applied for accurate prediction of software costs.

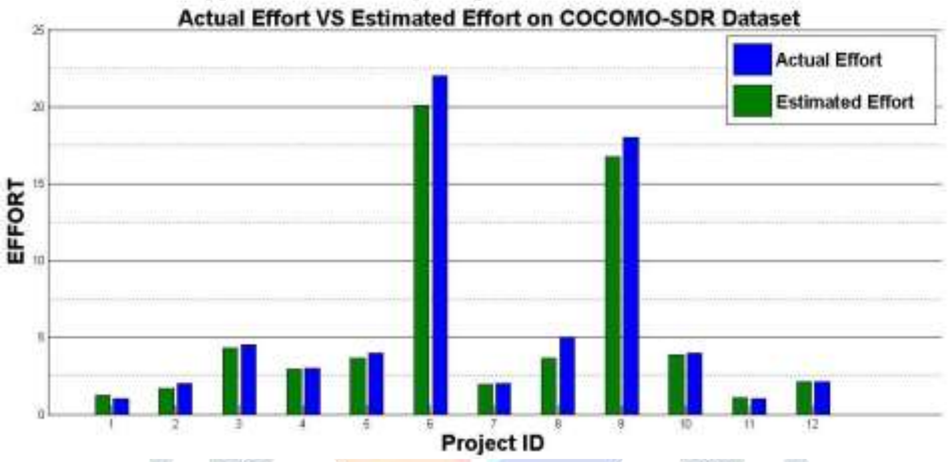

Figure 5- Comparison of Actual vs. Estimated effort on COCOMO_SDR Dataset

Conclusion and Future Work

In this work, we evaluated the harmony search based FLANN model for the task of Software cost prediction. The experiment has demonstrated that FLANN- harmony search performs the prediction task quite well. The result shows that the proposed harmony search algorithm can successfully train the FLANN for solving prediction problems with better accuracy under such vague available data. An important facet of this research work is to provide an alternative training scheme for training the Functional link Neural Network instead of standard BP learning algorithm for better Software Development Cost Estimations. The prime advantage of the proposed model is that it reduces the computational complexity without any sacrifice on its performance. The Results obtained from the proposed method have been evaluated using a no of criteria like MRE, MMRE and MdMRE. According to the obtained results it can be said that FLANN increases the accuracy of the cost estimation and especially when this model is trained using harmony search training algorithm the accuracy of the estimated cost became even better.

In the future, we can combine Meta-heuristic algorithms with ANNs and other data mining techniques to obtain better results in SCE field. In addition, the accuracy of SCE can be enhanced by applying other particle swarm optimization methods.

References

[1] Putnam, L.H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Vol. 4, No. 4, pp. 345 – 361, 1978.
[2] Boehm, B.W., (1981) Software Engineering Economics, Prentice Hall, Englewood Cliffs, NJ.
[3] J. C. Patra and R. N. Pal, "A functional link artificial neural network for adaptive channel equalization," Signal Processing, vol. 43, pp. 181-195, 1995.
[4] Jane M. Booker, Mary M. Meyer. ELICITATION AND ANALYSIS OF EXPERT JUDGMENT. Los Alamos National Laboratory.
[5] Rahul Reddy Nadikattu. 2017. The Supremacy of Artificial intelligence and Neural Networks. International Journal of Creative Research Thoughts, Volume 5, Issue 1, 950-954.
[6] JE Zull (2002). The Art of Changing the Brain: Enriching the Practice of Teaching by Exploring the Biology of Learning. Published: Stylus Publishing.
[7] M. Shepperd, M. Cartwright (2001), Predicting with Sparse Data, IEEE Trans. Soft. Eng. 27, 987-998

[7] Venkatachalam, A. R., “Software cost estimation using artificial neural networks”, in Proceedings of the 1993 International Joint Conference on Neural Networks, pp. 987-990, 1993.
[8] Rahul Reddy Nadikattu, 2014. Content analysis of American & Indian Comics on Instagram using Machine learning", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.2, Issue 3, pp.86-103.
[9] S. Dehuri and S.-B. Cho, "A comprehensive survey on functional link neural networks and an adaptive PSO–BP learning for CFLNN," Neural Computing & Applications vol. 19, pp. 187-205, 2010.
[10] Sikender Mohsienuddin Mohammad, "AN EXPLORATORY STUDY OF DEVOPS AND IT'S FUTURE IN THE UNITED STATES", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.4, Issue 4, pp.114-117, November-2016, Available at :http://www.ijcrt.org/papers/IJCRT1133462.pdf
[11] Y. H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," Computer, vol. 25, pp. 76-79, 1992.
[12] Pao, Y. H., Phillips, S. M., & Sobajic, D. J. (1992). Neural Net Computing and intelligent control systems. International Journal of Control, 56(2), 263–289.
[13] Sikender Mohsienuddin Mohammad, "AUTOMATION TESTING IN INFORMATION TECHNOLOGY", International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Volume.3, Issue 3, pp.118-125, August 2015, Available at :http://www.ijcrt.org/papers/IJCRT1133463.pdf
[14] Patra, J. C., Pal, R. N., Chatterji, B. N., & Panda, G. (1999). Identification of non-linear & dynamic system using functional link artificial neural network. IEEE Transactions on System, Man & Cybernetics – Part B; Cybernetics, 29(2), 254–262.
[15] I. A. Rastegar, “Development harmony search method for solving optimization problems: A Case Study in production scheduling parallel machines”, Journal of Industrial Engineering studies in production systems ,pp. 57-71,2013.
[16] www.promisedata.org