International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3890-3897
© Research India Publications. http://www.ripublication.com

An Efficient Software Effort Estimation by Combining Neural Network and Optimization Technique

[1]Azath. H, [2]Dr. P. Amudhavalli, [3]Dr. S. Rajalakshmi and [4]Dr. M. Marikannan

[1]Research Scholar, Karpagam Academy of Higher Education, Coimbatore, India.

Orcid Id: 0000-0002-7734-7745

[2]Research Supervisor & Professor, Department of Computer Science and Engineering, Karpagam Academy of Higher Education, Coimbatore, India.
Orcid Id: 0000-0001-6979-4847

[3]Professor & Head, Department of Computer Science and Engineering, Cheran College of Engineering, Karur, India.
Orcid Id: 0000-0003-4768-1029

[4]Professor, Department of Computer Science and Engineering, IRTT, Erode, India.
Orcid Id: 0000-0002-6983-1238

Abstract:

Estimating software efforts poses high challenge to both customers and developers. When the estimated effort is not precise, the management may be in menace of agreeing systems that may go beyond their budget, substandard with poorly developed functions and time consuming more than deadline. Hence, estimation of effort in an effective manner has become a significant task in the software development process, despite various difficulties ahead in doing so. The process also includes multiple steps to validate the minimized error. This paper introduces a novel approach for effort estimation based on diverse databases. The approach exploits neural network to aid in effort estimation with reduced cost and failure ratio. The neural network, which is presented here, is enhanced by incorporating optimization process. The optimization process selects weights for the neural network in such a way that improvement can be made on the classification model. In our method, the optimization process is carried out using Artificial Bee Colony (ABC) algorithm. The performance of the proposed model is investigated using parameters such as Mean Absolute Relative Error (MARE) and Mean Magnitude of Relative Error (MMRE). Performance comparison with existing models proves the competitive performance of the proposed model over the conventional models.

INTRODUCTION

The primary aim of software engineering can be portrayed as designing tools and techniques that aid in developing high quality, steady and maintainable applications. Several measurements are used by the developers and managers to validate and increase the quality of the application [7]. Any software product on openly available source components depends on many business and technical targets such as improved quality, short and firm development deadlines, limited development expenditures, source code accessibility, etc [3]. While evolution takes place in software, its structure has a primary impact on enhancement intensive locations, for instance, enhancing the implemented functionality [14, 15]. Software cost and efforts estimation is become a challenge for IT industries. There are lots of methods existing for efforts and cost estimation, but people do not know how to use these methods [22].

The software quality models must have the ability to identify crucial elements accurately. The elements that are determined as crucial must allow the application for specific certification tasks. The certification task may range from inspecting physically to static and dynamic analysis, testing and automatic formal analysis techniques. The software quality models also help in defining the reliability of the dispatched products. Numerous statistical models have been reported to estimate fault – proneness of program modules in software engineering [4].

Software project management is responsible for precisely envisaging the work – effort and duration to develop and/or manage a software system. The process is called software cost estimation [17], which is well-known for its significance on software project management. In order to make the managerial decision making process a successful phenomenon, estimation of software development effort must be accurate. Since the development cost is directly proportional to the project complexity, the early stage of development requires precise cost estimation [2]. It also acts as an essential component of infrastructure projects. Increased estimation accuracy helps the project managers to find alternatives and to get rid of underestimating technical and business solutions. When the project reaches the final stage, the accuracy of estimated cost will be high because of precise and comprehensive information at this stage [1].

The process of software cost estimation by analogy can be said as a prominent machine learning methodology. It can also be referred as a basic form of case based reasoning. The analogy based estimation is performed on the basis of assumption that like software projects it incurs the same project costs. Nevertheless, categorical variables on cost have to be handled by improved techniques [11]. Size, efforts to be put forth, development duration, adopted technology and ensured quality are the potential subjects of estimation in software development. Despite various effort models have been introduced, they consider software size as significant parameter, when the development effort has to be considered as the most significant subject. For instance, function point is a measurement model of software size. It exploits logical function terms that make both users and owners understand easily. However, the measured size remains constant, because it considers only the functional requirements irrespective of the contribution of programming language, design technology and development skills [13].

The cost estimation process may face the following problems.

- Difficulty ahead in mapping cost – affecting factors with output metrics, because these factors exhibit nonlinearity
- Difficulties in measuring the metrics because of incompleteness and imprecision of data acquired at initial stage
- Though numerous models prevail for estimation, challenges remain in identifying suitable model for the current circumstance and
- Difficulties in combined usage of algorithmic and non – algorithmic models [12].

Nowadays, the community is highly relying on information technology and software systems. This necessitates an appropriate estimation method as it has a wide impact on project cost and quality. Numerous estimation models have been reported in the literature, where more researchers sill intend to formulate better estimation models [18]. The rest of the paper is organized as follows. Section 2 presents a review report on recent research works. Section 3 details the proposed methodology, which includes effort estimation using neural network and optimization process. Section 4 discusses the results of the proposed technique.

RELATED RESEARCHES

The focus on using soft computing techniques for estimating software effort has been significantly increased through various research works. This Section presents the review on few of the methodologies.

Rao *et al* [5] have estimated cost using a computationally efficient Functional Link Artificial Neural Network (FLANN). By reducing the computational complexity, the neural net can be used for online applications. The network has exhibited a simple architecture without any full back propagation training and hidden layer. The adaptive neural network has worked efficiently by firstly using COCOMO approach for estimating the software cost and then by using FLANN with backward propagation. The entire network has remained as “white box” because of its comprehensive analysis on every neuron. The simple architecture and appropriate training by back propagation algorithm has ensured better accuracy than other methods.

Reddy *et al.* [6] have introduced a model to estimate the software effort. The model was based on artificial neural network and designed in such a way that the network performance can be adequately improved so as to meet the COCOMO model requirements. Multilayer feed forward neural network has been used for the model and its parameters, which has aided in estimating the software development effort. Back propagation learning has trained the network by iterative comparison of actual effort and network’s prediction, when a training set has been given as input. The neural network model has outperformed over other model in terms of estimation accuracy, when experimentation has been carried out on COCOMO dataset.

Though various research eras have crossed, no attempt has been made to identify the most established software effort estimation methods to ensure superior accuracy. Moreover, it has been reported in the literature that there is no consistent performance by any of M estimation methods that have been considered for study. Certain assertions have been made to develop ensembles of various estimation methods rather than identifying best estimation method. Ekrem et al. [8] have introduced a method, which has unified nine learners with ten preprocessing techniques so that 9 10 ¼ 90 solo methods can be developed. Experimentation has been carried out using 20 datasets and investigation has been done using seven error measurements. The experimental outcome has revealed top n (here, n ¼ 13) individual methods, which have exhibited performance stability on varying datasets and error measurements. The obtained best 2, 4, 8, and 13 individual methods have been unified to produce 12 multi – methods that have been further subjected to compare with the individual methods.

Given a small significant content, there should be a concise information and minimal value added to the complex learning schemes. Ekrem et al. [9] have introduced a QUICK method, which has determined the Euclidean distance between instances (rows) and features (columns) of SEE data. Further, it has pruned synonyms and outliers, which are similar features and distant instances, respectively, followed by evaluating the reduced data. The data reduction has been done by comparing predictions that have been obtained from (1) a simple learner and (2) a state – of – the – art learner (CART), which used reduced data and collective data, respectively. Performance investigation has been carried in hold – out experiments using mean and median MRE, MAR, PRED and MBRE.

Since software cost estimation process estimates the amount of effort and time to put forth to develop a software system, it is considered as the most significant process. It is also considered as a crucial task. A precise estimate can provide a strong fundamental for the development procedure. Anupama et al [10] have explained Constructive Cost Model (COCOMO), which is a renowned software cost estimation model. Artificial neural network with perceptron learning algorithm has facilitated the implementation of the model. The training and validation of the network have been carried out using COCOMO dataset.

In the recent era, effort to put forth in a software project is the most analyzed variable and estimating the effort has been found as the most tedious task in a project management process. However, a software cost estimation technique facilitates this task by estimating the amount of effort to put forth and development duration to construct the software system. It has been considered as a significant task to help the software sectors in managing the development process proficiently. Numerous cost estimation models have been reported in the literature, where each model has portrayed its own merits and demerits in envisaging the development cost and effort. Anupama et al. [16] have worked out on estimating software cost using back propagation neural networks. The developed model can cope up with renowned COCOMO model and work for its performance improvement. It has the ability to handle imprecision and uncertainty of the input and to improve the reliability of the estimated software cost. The experimentation of the model has been carried out on three openly available software development datasets.

The software cost estimation process is a difficult and time consuming task. Analogy – based estimation of software effort has been claimed as one of the viable techniques in the field. However, the method remains incapable in a circumstance of precise handling of categorical data. Initially, the software effort estimation models were developed based on regression analysis and mathematical models. But nowadays, the models are on the basis of simulation, soft computing, genetic algorithm, neural network, fuzzy logic modeling, etc. Ziauddin et al. [17] have attempted to increase the software effort estimation accuracy by exploiting fuzzy logic model. The approach has fuzzified the input variables of COCOMO II model and defuzzified the output variable to obtain the estimated effort. The linguistic terms for COCOMO II model have been defined using triangular fuzzy membership functions.

Software development effort estimation was considered a fundamental task for software development life cycle as well as for managing project cost, time and quality. Therefore, accurate estimation was a substantial factor in projects success and reducing the risks. In recent years, software effort estimation has received a considerable amount of attention from researchers and became a challenge for software industry. In the last two decades, many researchers and practitioners proposed statistical and machine learning-based models for software effort estimation. Ghatasheh [23] have proposed Firefly Algorithm as a metaheuristic optimization method for optimizing the parameters of three COCOMO-based models. These models include the basic COCOMO model and other two models proposed in the literature as extensions of the basic COCOMO model. The developed estimation models are evaluated using different evaluation metrics.

PROPOSED METHOD FOR SOFTWARE EFFORT ESTIMATION

Software designing faces a significant task, called software effort estimation, which is a process of estimating the effort to put forth to construct a software system. However, the process of software effort estimation is a tedious task and hence it remains as a potential research platform since past few decades. The cost and time estimates can be used for initial coarse validation and for monitoring the progress of project development at the development stage, whereas, they can be used for evaluating the productivity of the project at the final stage. As a whole, effort estimation aids in developing fulfilled proficient software. Our method exploits soft computing for estimating the software effort, where classification is performed using neural network. The weights of the neural network are optimized using optimization algorithm.

A.Steps involved in the Effort Estimation Process

Generally, various software parameters are used by the estimation process to determine the effort to put forth for developing the particular software. Figure 1 portrays the steps that constitute the proposed effort estimation process. The dataset is comprised of numerous parameters that are used to determine the actual and estimated efforts followed by calculating MRE value. Consequently, these values are classified by applying classification process for which neural network is used. It helps to identify the suitable parameter value to perform the process. In our method, the neural network is enhanced by optimizing the weights using optimization algorithm. Here, ABC is used to solve the purpose. Unifying ABC here helps in improving the classification process. Eventually, an effort value from the classifier is obtained and it is considered as the near – best effort value with reduced error rates.

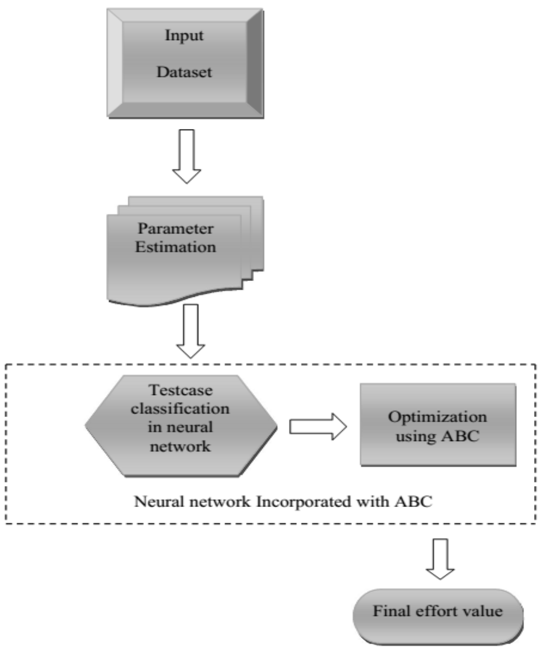
Input
Dataset
Parameter Estimation
Testcase classification in neural network
Optimization using ABC
Neural network Incorporated with ABC
Final effort value

Figure 1. Proposed software effort estimation model.

The dataset is comprised of numerous parameter values that have to be subjected to classification. Hence, these values are applied to the neural network for classification of relevant parameters. The following section details the classification process of neural network.

B. Training in Neural Network

The software parameter values are acquired from the dataset and given as input to the neural network classifier to perform classification. The general training of neural network enable mapping the input to a specific output. The neural network is highly compatible with the classification process. The feed forward neural network is exploited in the training phase of the proposed method. Comparison is made between the parameter values and the input files followed by classifying the suitable parameters. A neural network is comprised of three layers namely, input layer, hidden layer and output layer. The basic architecture of the feed forward neural network can be observed from Figure 2.

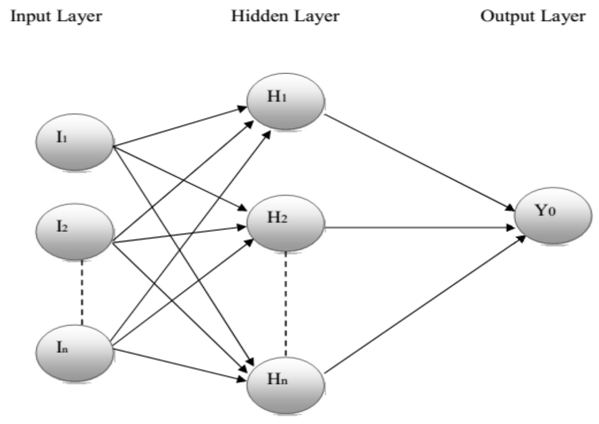

Figure 2. General Feed Forward Neural Network Architecture.

ABC, which is an optimization algorithm, works in the proposed method to allocate weights on different nodes of the neural network. In other words, ABC determines relative weights.

Proposed Artificial Bee Colony for Optimization of weights in Neural Network:

In an ABC model, a food source position refers to a possible solution for the optimization problem, whereas nectar quantity of a food source refers to the quality (fitness) of the respective solution. The objective of bees is set as to determine the best solution [20]. Each employed bee shares the information with an onlooker bee and flies back to the food source, which was visited by it in the previous wandering. This process is undertaken since the memory keeps the record of the food source. The employed bee selects a new food source using the visual knowledge about the neighborhood of the one stored in the memory and assesses the nectar amount [19].

Employee Bee Phase

There are three bee groups in the artificial bee colony. They are employed bees, onlooker bees and scout bees. An employed bee is a bee that flies back to its previously visited food source, whereas an onlooker bee is a bee resides in the dancing area to decide on the food source to be selected. Scout bees are bees that undergo random wandering for food source. The employed bees contribute the first half of the colony, whereas the rest is contributed by onlooker bees. Each employed bee contributes by a food source. In other words, the number of food sources around the hive and the number of employed bees is same.

Initially, the employed bees select random set of food source positions for which the nectar amounts are calculated. They reach the hive and distribute the nectar information of the food sources to the onlooker bees, which are in the dancing region of the hive. In the perspective of algorithm, arbitrary set of initial population p_i with n solutions, where each solution is the food source position and S_p is the population size is generated. The solution representation can be given as $h_i, where\ 1 \le i \le n$ is an N-dimensional vector and N is the number of parameters to be optimized. Once the population is initialized, it is subjected to the iterative process that involves employed bees, onlooker bees and scout bees.

Onlooker Bee Phase

This phase enables the onlooker bees to select the food sources based on the nectar information obtained from the employed bees and to generate new set of solutions. Generally, the onlooker bee is font of food source area, which has substantial nectar information shared by the employed bees in the dancing region of the hive. The probability of selecting a food source by an onlooker bee is directly proportional to the nectar amount of the food source. Hence, the employed bee dancing with higher nectar value assigns an onlooker bee for the food position. The probability of selecting a food source (F_s) by an onlooker bee can be given as follows.

$$F_s = \frac{f_i}{\sum_{k=1}^{n} f_k} \tag{1}$$

where,

f_i refers to fitness of the solution and

n refers to the number of food sources which is equal to the number of employed bees.

The onlooker bee reaches the selected food source and selects a new neighborhood of the selected food source based on visual knowledge. The visual knowledge is obtained by comparing both the food positions. When the bee abandons a food source due to its lesser nectar value, the scout bee generates a new random food source and fills the abandoned position. The onlooker bee modifies the food source position stored in the memory and hence finds new food source and validates the nectar amount (fitness) of it.

If we consider an old position $g_{i,k}$ and a new position $h_{i,k}$, the relationship can be given as

$$h_{i,k} = g_{i,k} + \rho_{i,k}(g_{i,k} - g_{j,k}),\ i \neq j \tag{2}$$

Where,

$j = \{1,2,...,n\}$

$k = \{1,2,...,N\}$

$\rho_{i,k}$ is an arbitrary number in the range [−1, 1].

The position update equation interprets that a decrease in the deviation between the parameters of $g_{i,k}$ and $g_{j,k}$ leads to a decrease in the perturbation on the position $g_{i,k}$. Hence adaptive reduction in the step length happens, when optimal solution in the search space has reached. Reformulating the position updating step leads to the following equation.

$$h_{i,k} - g_{i,k} = \rho_{i,k}(g_{i,k} - g_{j,k}) \tag{3}$$

A time domain representation can be given for the position update equation by considering $g_{i,k}$ as X_l when $h_{i,k}$ is taken as X_{l+1}. Hence, we obtain

$$X_{l+1} - X_l = \rho_{i,k}(g_{i,k} - g_{j,k}) \tag{4}$$

As $X_{l+1} - X_l$ refers to discrete version of the derivative of order $\alpha = 1$, we can write

$$D^{\alpha}[X_{l+1}] = \rho_{i,k}(g_{i,k} - g_{j,k}) \tag{5}$$

Scout Bee phase

The scout bees are the employed bees whose food sources are abandoned by the employed and onlooker bees. They search randomly and replace the abandoned food sources by new food sources. This process can be simulated by replacing the abandoned solution by randomly generated solution. A food source is said to be abandoned, when the position does not provide any improvements over a pre-defined number of iterations, often termed as limits. A typical ABC algorithm enables the scout bees to search the solutions arbitrarily within the vicinity of the hive. This style of search may be advantageous at the beginning stages of iterations, but it may fail at the final stage of iterations. Hence, global search is recommended for scout bee at the initial stage and local search at the final stage of iterations. As there may not be any improvement even from the best food source at the final iterations, the scout bees are selected and removed from the population. Hence, ABC works out to determine the suitable weight for each network node to increase the classification performance.

RESULT AND DISCUSSION

The proposed method is implemented in Netbeans 7.4, which is a renowned platform to use for effort estimation process because of its compatibility. The platform offers variety of applications that can be developed from a set of components, which are often referred as modules. As Netbeans has the collective set of modules as built – in functions to develop program, it is convenient for any user to initiate the work immediately. The study considers Desharnais dataset, which has 81 projects with incompleteness in 4 projects that can be removed. There are nine independent variables and one dependant variable in the dataset.

Despite numerous error measures are in practice, the most renowned error measure is Mean Absolute Relative Error (MARE).

$$MARE = \sum_{e=1}^{n}(|(E_e - A_e)/A_e|) \tag{6}$$

where,

E_e - Estimated effort

A_e -Actual effort

A. Experimental Results

The obtained experimental outcomes from the proposed method are tabulated in Table 1. The actual effort and the estimated efforts are determined for various sizes followed by determining the Magnitude of Relative Error (MRE) for each entry. The actual effort is typically lesser than the estimated effort. As per the equation (8) given below in the performance evaluation section, MMRE for the efforts are determined for the execution time

Table 1. Effort Estimates and MRE

No	Original effort	Estimated effort	MRE
1	7538.342	2199.522	0.708222
2	5037.401	2357.995	0.531902
3	797.177	380.125	0.523161
4	4926.516	1696.964	0.655545
5	3341.558	996.325	0.701838
6	4195.771	1207.623	0.712181
7	3157.767	1095.779	0.652989
8	5510.152	1661.626	0.698443
9	4817.673	3108.069	0.354861
10	2445.469	995.483	0.592928
11	5735.558	1730.208	0.698337
12	7249.34	3546.582	0.510772
13	1452.7	913.559	0.37113
14	5002.478	1745.231	0.651127
15	5674.268	2139.205	0.622999
16	3098.382	775.079	0.749844
17	4038.353	1269.224	0.685708
18	7007.879	1530.824	0.781557
19	7885.784	1900.384	0.759011
20	1633.202	399.754	0.755233
21	9488.004	5987.829	0.368905
22	8337.024	2111.167	0.746772
23	7422.679	2544.948	0.657139
24	7847.99	4252.522	0.458139
25	7045.08	1694.464	0.759483
26	6071.397	1417.001	0.76661
27	5218.878	1544.466	0.704062
28	8758.911	1971.126	0.774958
29	6666.486	2881.585	0.567751
30	7795.007	1842.441	0.763638

B. Performance Analysis

The equations that are given below are used to determine MRE and MMRE.

$$MRE = \left|(A_e - E_e)\right| / A_e \tag{7}$$

where,

E_e - Estimated effort

A_e -Actual effort.

The MMRE calculation for the estimated effort can be done using equation (7). The proposed method has accomplished better MMRE than other fuzzy – based works.

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{8}$$

where n= number of data * attributes taken

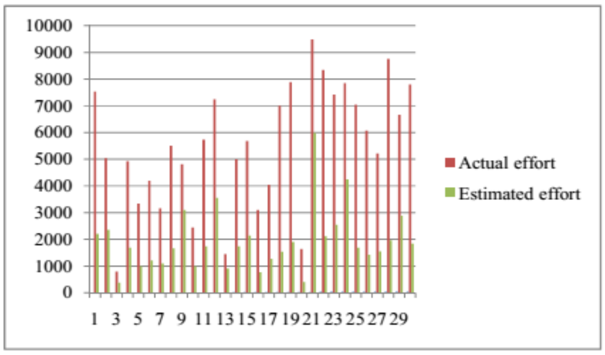

Figure 4. Graphical representation for Effort Estimates

The graphical illustration of the obtained effort value from the proposed method is affixed in Figure 4 in which the actual effort is relatively higher than the estimated effort values.

The accomplished improvement in effort estimation is observed using MMRE and MARE of the proposed method prior optimization and post optimization process. The obtained values are tabulated below.

Table 2. Parameter values before and after applying ABC algorithm for optimization.

Parameters	**Proposed Results in (%)**	
	Before Optimization	**After Optimization**
MMRE	13.608	0.6428
MARE	65.639	19.28524

The performance improvement accomplished from post – optimization process is better illustrated using the following figure, where the error values obtained from prior optimization and post optimization are plotted.

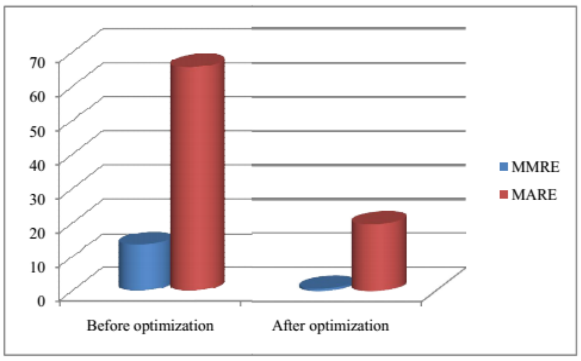

Figure 5. Parameter values before and after applying ABC algorithm for optimization

The effectiveness of the proposed method is then proved by compairing it with existing methods through MMRE measurements obtained from proposed and existing methods. The values are tabulated in Table 2, where MMRE is given in percentage.

Table 3. MMRE measurements for the proposed and existing methods.

METHODS	**MMRE (%)**
Proposed Method	0.6428
Fuzzy method	30. 6
Analogy with fuzzy number	26.89

Figure 6 portrays the graphical illustration of comparative results between the proposed method and the existing method based on MRE and MMRE measurements. Here, the comparison interms of MMRE is made between the proposed system and the existing methods given in [21]. The graphical illustration demonstrates that the proposed method outperforms the existing methods in terms of efficiency.

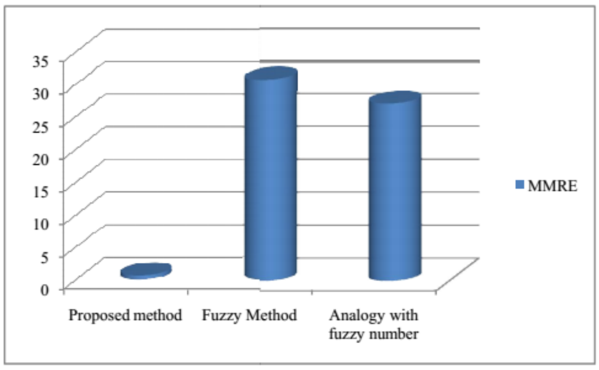

Figure 6. The comparison of MMRE measure for the proposed method

CONCLUSION

Designing a software system requires software effort estimation significantly. Numerous research works have been carried out to increase the precision of effort estimate of the software system. This paper has proposed a novel approach to estimate the software effort precisely. The approach has been contributed by neural network classification process and an optimization process. The neural network has classified various software parameters. For betterment of classification performance, ABC has been used to optimize the weights of neural network. Error parameters such as MRE, MMRE and MARE have been determined and performance comparison has been made with the existing method. The experimental outcomes have demonstrated the proposed system outperform the existing method in estimating the software effort more precisely.

REFERENCE

[1] Jamshid Sodikov, "Cost Estimation of Highway Projects in Developing Countries: Artificial Neural Network Approach", Journal of the Eastern Asia Society for Transportation Studies, Vol. 6, pp.1036-1047, 2005.

[2] Ch. Satyananda Reddy and KVSVN Raju, "An Optimal Neural Network Model for Software Effort Estimation" Int. J. of Software Engineering, IJSE Vol.3, No.1,2010.

[3] Heikki Orsila, Jaco Geldenhuys, Anna Ruokonen and Imed Hammouda, "Update Propagation Practices in Highly Reusable Open Source Components", In. proc.of 20th World Computer Congress on Open Source Software, Milano, Italy, Vol. 275, pp.159-170, Sep 7-10, 2008.

[4] Yue Jiang, Bojan Cukic and Yan Ma, “Techniques for Evaluating Fault prediction models”, Springer Journal of Software Engineering, pp. 561–595, 2008.

[5] B. Tirimula Rao, B. Sameet, G. KiranSwathi, K. Vikram Gupta, Ch. Ravi Teja and S. Sumana, "A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN)", International Journal of Computer Science and Network Security, VOL.9 No.6,2009.

[6] Ch. Satyananda Reddy and KVSVN Raju, "A Concise Neural Network Model for Estimating Software Effort", International Journal of Recent Trends in Engineering, Issue. 1, Vol. 1,2009.

[7] Jehad Al Dallal, "Mathematical Validation of Object-Oriented Class Cohesion Metrics", International Journal of Computers, Vol. 4, No.2, 2010.

[8] Ekrem Kocaguneli, Tim Menzies, and Jacky W. Keung, "On the Value of Ensemble Effort Estimation", IEEE transactions on software engineering, Vol. 38, No. 6, 2012.

[9] Ekrem Kocaguneli, Tim Menzies, Jacky Keung, David Cok and Ray Madachy, "Active Learning and Effort Estimation:Finding the Essential Content of Software Effort Estimation Data", IEEE Transactions on Software Engineering, Vol.39, No.8,2013.

[10] Anupama Kaushik, Ashish Chauhan, Deepak Mittal and Sachin Gupta, "COCOMO Estimates Using Neural Networks", I.J. Intelligent Systems and Applications, Vol.9, pp.22-28, 2012.

[11] S.Malathi and Dr.S.Sridhar, "Estimation of Effort in Software Cost Analysis For Heterogenous Dataset Using Fuzzy Analogy, “International Journal of Computer Science and Information Security,Vol.10, No.10, 2012.

[12] Divya Kashyap, Ashish Tripathi and Prof. A. K. Misra, "Software Development Effort and Cost Estimation: Neuro-Fuzzy Model", Journal of Computer Engineering, Vol.2, No.4, pp.12-14, 2012.

[13] Mohammad Saber Iraji and Homayun Motameni, "Object Oriented Software Effort Estimate with Adaptive Neuro Fuzzy use Case Size Point (ANFUSP)", I.J. Intelligent Systems and Applications, Vol.6, pp. 14-24,2012

[14] Parag C. Pendharkar, "Probabilistic estimation of software size and effort", Expert Systems with Applications, Vol. 37, pp.4435–4440,2010.

[15] Andrzej Olszak, Eric Bouwers, Bo Nrregaard Jrgensen and JoostVisser, "Detection of Seed Methods for Quantification of Feature Confinement", Inproc.of the 50th International Conference on Objects, Models, Components, Patterns, 2012.

[16] Anupama Kaushik, A.K. Soni and Rachna Soni, "A Simple Neural Network Approach to Software Cost Estimation", Global Journal of Computer Science and Technology Neural & Artificial Intelligence, Vol.13, No.1, 2013.

[17] Ziauddin, Shahid Kamal, Shafiullah khan and Jamal Abdul Nasir, "A Fuzzy Logic Based Software Cost Estimation Model", International Journal of Software Engineering and Its Applications, Vol. 7, No. 2,2013.

[18] Ihtiram Raza Khan and Prof. M. Afshar Alam, "Software cost estimation using a Neuro-Fuzzy algorithmic approach", International Journal of Computer Science and Management Research, Vol. 2 No.7,2013.

[19] Dervis Karaboga, Bahriye Akay, A comparative study of Artificial Bee Colony algorithm, Journal of Applied Mathematics and Computation, Vol. 214, Pp. 108–132, 2009.

[20] Dervis Karaboga and Celal Ozturk, Fuzzy clustering with artificial bee colony algorithm, Journal of Scientific Research and Essays, Vol. 5, No. 14, pp. 1899-1902, 2010.

[21] S.Malathi and Dr.S.Sridhar, "Estimation Of Effort In Software Cost Analysis For Heterogenous Dataset Using Fuzzy Analogy",International Journal of Computer Science and Information Security,Vol.10, No.10,2012.

[22] Poonam Rijwani, Sonal Jain and Dharmesh Santani, "Software Effort Estimation: A Comparison Based Perspective", International Journal of Application or Innovation in Engineering & Management, Vol.3, No.12, 2014.

[23] Nazeeh Ghatasheh, Hossam Faris, Ibrahim Aljarah and Rizik M. H. Al-Sayyed, "Optimizing Software Effort Estimation Models Using Firefly Algorithm”, Journal of Software Engineering and Applications, Vol.8, pp.133-142,2015.