International Journal of Electrical and Computer Engineering (IJECE)
Vol. 11, No. 1, February 2021, pp. 707~715
ISSN: 2088-8708, DOI: 10.11591/ijece.v11i1.pp707-715

Software development effort estimation modeling using a combination of fuzzy-neural network and differential evolution algorithm

Amir Karimi[1], Taghi Javdani Gandomani[2]
[1]Department of Computer Engineering, Isfahan Branch, Islamic Azad University, Iran
[2]Department of Computer Science, Shahrekord University, Iran

Article Info

Article history:

Received Mar 5, 2020
Revised Jun 20, 2020
Accepted Aug 16, 2020

Keywords:

Differential evolution algorithm
Effort estimation
Neural network
Neuro-fuzzy inference system
Software development

ABSTRACT

Software cost estimation has always been a serious challenge lying ahead of software teams that should be considered in the early phases of a software project. Lack of sufficient information on final requirements, as well as the existence of inaccurate and vague requirements, are among the main reasons for unreliable estimations in this area. Though several effort and cost estimation techniques and models have been proposed over the recent years, an increase in their accuracy has always been a controversial issue, and researchers' efforts in this area are still ongoing. This study presents a new model based on a hybrid of adaptive network-based fuzzy inference system (ANFIS) and differential evolution (DE) algorithm. This model tries to obtain a more accurate estimation of software development effort that is capable of presenting a better estimate within software projects compared to previous works. The proposed method outperformed other optimization algorithms adopted from the genetic algorithm, evolutionary algorithms, meta-heuristic algorithms, and neuro-fuzzy based optimization algorithms, and could improve the accuracy using MMRE and PRED (0.25) criteria up to 7%.

Corresponding Author:

Taghi Javdani Gandomani,
Department of Computer Science,
Shahrekord University,
Rahbar Blvd, Shahrekord, Iran.
Email: javdani@sku.ac.ir

1. INTRODUCTION

Software project planning and management are regarded as essential activities in software product development. Accurate estimation is one of the key activities in project planning to complete a project successfully. In software project management, estimation is required at the beginning of the process as one of the important phases of software development, including cost, time, and effort estimation [1]. Often cost estimation for software projects is the estimation of the effort required to implement these projects.

Effort estimation is the prediction of the amount of effort in terms of time and the number of the team members required to develop the software product because one of the cost elements in software projects is the cost of training and payments to employees. Proper estimation can include the accurate estimation of the software costs in budgeting, planning, controlling, and finally managing the project effectively. Moreover, effort estimation can be helpful in determining the required resources in the future [2]. Software development cost estimation has always been proposed as a challenge ahead of software teams and companies. Accurate and reliable effort estimation is essential for resource allocation and reasonable planning during software design and development. Success and failure of a project depend on the project

planning because the time and budget constraints required for the successful completion of the project are estimated in this phase [3].

Inadequate estimation of the required effort may cause delay and cost overrun, leading to project failure, and conversely, overestimation can be detrimental to using project resources. Therefore, it seems that the success of any software project seriously depends on the accuracy of its effort estimation [4]. Accurate estimation is helpful in contract negotiations, planning and coordinating project activities, and allocating effective resources to them. The notion of software cost estimation began to receive attention with the advent of the computer industry in 1940, and research is still ongoing in this area [5]. Over the past decade, various types of development processes have been established, which had different effects on software project planning and estimation. How to generate estimation to achieve better results has long been the focus of the software engineering industry [6].

Following the achievement of higher levels of accuracy in effort estimation, the experiments have been continuously investigated in computational techniques using theoretical concepts [7, 8] and the combination of existing methods [9-11]. Considering the the difficulty of analyzing the relationship between properties and the complexity of effort estimation, the optimization process should be applied. Optimization can be applied either directly to effort estimation processes such as feature weighing in similarity-based estimation or indirectly using machine learning techniques like, e.g., fuzzy neural networks or artificial neural networks.

The purpose of the present paper is to model the effort estimation required for software development using the hybrid of fuzzy neural network and differential evolution (DE) algorithm. The rest of the paper is organized as follows: Section 2 provides a concise review of the related works. The proposed model, along with the research process, is presented in Section 3. Section 4 explains the results obtained from this research. Finally, the conclusion is presented in Section 5

2. RELATED WORKS

In this section, research studies on software development effort estimation are reviewed by investigating fuzzy and neural systems-based, and machine learning-based research studies.

2.1. Machine learning methods and optimization algorithms in software development effort estimation

This section gives an overview of studies that have used machine learning methods and optimization algorithms for software development effort estimation. In 2013, Khatibi *et al*., [12] used particle swarm optimization (PSO) algorithm to weigh features in the similarity-based method. This hybrid led to the identification of similar projects which are based on the similarity function in a similarity-based method. They proposed a new framework in which more accurate estimations are obtained by weighing features properly. As this framework can be used in the datasets with batched and non-batched features, it can be said that it is flexible to a large extent. In this study, three real datasets were used to assess the proposed model, and the results were compared with other estimation models. The results show that this model can significantly improve the performance of existing estimation models [12].

In 2015, Kumari *et al*., [13] used the PSO algorithm to initialize parameters of the use case point analysis method. They believe that meta-heuristic techniques can be used through computational methods to develop test effort estimation models to optimize the problem by repeatedly trying to improve a solution. Therefore, in this research, a search meta-heuristic algorithm was used. The results show that this accuracy implementation increases the precision of effort estimation.

Hota *et al*., [14] studied adaptive neural networks in 2015. To this purpose, they adjusted some particular parameters such as momentum, learning rate, error backpropagation network (EBPN), and implemented this method using two datasets: China and Maxwell. The analyzed results from various surveys were satisfactory.

In another research in 2016, Rijwani *et al*., [15] investigated the feed-forward neural network. They used the COCOMO dataset to test and train network in this research. Additionally, they used mean square error (MSE) and mean magnitude of relative error (MMRE) as performance measurement indices. The results obtained from this research show that the proposed model can provide better results and a more accurate prediction of software development efforts.

By assessing development efforts in the initial stages through a popular machine learning method called random forest technique in 2016, Satapathy *et al*., [16] adjusted parameters in use case point analysis method. Using this method, the effort parameters were optimized to obtain higher accuracy. Moreover, the obtained results were compared with multilayer neural networks, radian-based networks, linear and nonlinear regression methods, and the performance of each method is obtained.

Nassif *et al*., [17] investigated neural network models for software development effort estimation. They investigated four different types of neural networks, including radial basis function neural networks,

general regression neural network, cascade correlation neural networks, and multilayer perceptron networks. ISBSG dataset was used to test and train these four models in this research. The obtained results show that the four models tend to overestimate in 80% of the datasets, and the significance of the model inputs differs depending on the model chosen. Finally, it was found that the cascade correlation neural network outperforms the other three models [17].

Benala *et al*., [18] used the DE algorithm to weigh features in the similarity-based method. To this purpose, they used five successful mutation strategies. They simulated the proposed method on the PROMISE repository dataset and obtained a significant improvement in prediction over-optimization of features weighing in other methods such as artificial neural network, genetic algorithm, PSO algorithm, and radian-based networks.

2.2. Fuzzy and neural systems in software effort estimation

In 2010, Vishal *et al*., [19] proposed an optimal fuzzy logic-based framework to predict software development efforts. This framework proposes transparency in the system, and with the availability of new data, can adapt to changing environments. The proposed traditional COCOMO cost estimation model tried to improve the accuracy of effort estimation using the fuzzy concept with measuring development state for projects and cost stimuli.

Lopez-Martin [20] proposed a fuzzy model for effort prediction in small scale programs based on two independent variables. He used the recode of small programs collected by 74 programmers using personal software process methods. These data were used as input in a fuzzy model for development effort estimation. The accuracy of this fuzzy model was compared with that of a statistical regression model. The comparison criterion was MMREMMRE. In the verification and validation stages, it was found that MMRE value in the fuzzy model is lower than or equal to that of the regression model, and in comparison to ANOVA-based models, no significant statistical difference was observed between their instruments. This conclusion shows that fuzzy logic can be used for effort prediction in small programs [20].

In 2013, Kamal *et al*., [21] proposed a fuzzy-based model for software cost estimation. In this method, fuzzy logic was used to compress the input parameters of the COCOMO II model. Triangular fuzzy numbers were used to express linguistic terms in the COCOMO II model. The results from this model were compared with those of COCOMO II and Alaa Sheta. The proposed model revealed better results in terms of MMRE, PRED (n), and CSF.

Kad *et al*., [22] used a soft computation-based technique to investigate uncertainty and unambiguity problems in 2013. This technique led to the improvement of the software development efforts estimation. In this vein, fuzzy logic was applied to different parameters of the COCOMO II model. The obtained results showed that the MMRE value and values obtained using fuzzy logic were better than that of the algorithm model. Results were obtained using the COCOMO dataset.

In 2015, Agrawal *et al*., [23] investigated the efficiency of software development effort estimation using the fuzzy neural model. In this research, the fuzzy neural model with three different membership functions is compared with neural network models. The fuzzy neural model with triangular, Gaussian, and Bayesian membership function was compared with the neural network model. To this purpose, Lopez-Martin dataset [20] with 41 modules was used, and three different membership function models were compared with the existing neural network models based on five different parameters, including magnitude relative error (MRE), MMRE, prediction (PRED), balanced relative error (BRE), relative standard deviation (RSD) and root mean squared error (RMSE). Finally, the comparison showed that the neuro-fuzzy model using a trapezoidal membership function outperforms other models. Additionally, it was found that the neuro-fuzzy model using a triangular membership function gives better results for all the five parameters [23].

In 2016, Nanda *et al*., [24] used the hybrid PSO algorithm and fuzzy neural network model to improve results. They used NASA datasets. The parameter cost driver consists of 17 features that have been optimized using PSO techniques to achieve better prediction accuracy. Moreover, the optimization results were trained using the algorithm to get a prediction neuro-fuzzy effort. The performance of the proposed estimation model was assessed with some parameters of the intelligent system using MSE, MMRE, and right PRED criteria. The results from this research showed the lower and higher rate of MSE and MMRE and PRED, respectively.

Moosavi *et al*., [25] proposed a novel hybrid of adaptive fuzzy inference system (ANFIS) and the satin bowerbird optimization (SBO) algorithm to obtain more accurate software development efforts estimation. The proposed hybrid model is an optimized neuro-fuzzy-based estimation model that is able to produce accurate estimation in software projects. In comparison to other optimization algorithms, the proposed optimization algorithm is inspired by the genetic algorithm using 13 standard test sequences, including multimodal and unimodal functions. Also, the proposed hybrid model was assessed using three datasets. The results from this research showed that the presented model could improve performance criteria. This paper was used as the base paper in this research

3. THE PROPOSED MODEL

In this research, the software effort estimation model called ANFIS-DE is proposed. This model is presented in two sections. In the first section known as the training phase, how to obtain the optimal parameter for ANFIS is investigated using the DE algorithm. In the second phase known as the test phase, we discuss how to use the obtained optimal parameters. In fact, a set of training data is presented to the model for the first time, and after the parameters were extracted from the fuzzy basis system, the software effort estimation system is adjusted by the parameters, and the DE algorithm investigates the parameter vector in training samples to minimize errors. To obtain the optimal parameters, the stages to design and create an optimized neuro-fuzzy system can be classified as follows:

- Receiving datasets consisting of software projects with identical features;
- Creating a fuzzy basis system;
- Adjusting fuzzy system parameters using modeling error with the DE algorithm;
- Returning a fuzzy system with the best parametric values as the final result.

In the present research, a set of software projects is used for effort prediction as a training dataset in the training phase. First, a Sugeno type fuzzy system is generated, and data are injected into it. Then, the fuzzy system parameters are set. This is the most important stage of ANFIS optimal design. The process of setting parameters can be described as follows:

- Reading fuzzy basis system parameters: this includes reading input and output membership function parameters. After reading these parameters, they should be sequentially saved in a vector.
- Changing the fuzzy basis system parameters using the DE algorithm:

To this purpose, we suppose that the value of the optimal parameter is a factor of the initial parameters. In other words, if we suppose that p_i* is the optimal value of parameter i, p_i^0 is the initial value of parameter i, and x_i is the coefficient of this state, p_i*is obtained using the (1).

$$p_i^* = x_i p_i^0 \quad (1)$$

where x_i is determined using the DE algorithm. There are two opinions about xi:

First: just resizing without changing signs. $x_i \epsilon [10-\alpha, 10\alpha]$ and the optimal value of α can be less than or equal to 1.

Second: Resizing and changing signs. $x_i \epsilon$ [-M, +M] where M can be less than or equal to 10.

- Entering new parameters obtained from the optimization algorithm in the fuzzy basis system
- Classifying data and deducing the obtained results for effort estimation
- Calculating effort using system error assessment criteria, including MMRE and PRED (0.25)

At this stage, the best parameters obtained by DE are saved for the next stage. These parameters were obtained based on MMRE and PRED (0.25) criteria. In fact, DE calculates these parameters to minimize the value of MMRE and PRED (0.25). Generally, the main aim of all software effort estimation techniques is to reduce MMRE and PRED (0.25).

The second phase, which is called the test phase, is the proposed model assessment stage. In fact, MMRE and PRED (0.25) values are the outputs of this stage. Like the previous stage, part of the software project is selected as test data and applied to the fussy basis system. The final values of MMRE and PRED (0.25) are calculated after the optimal parameters are inserted. In the proposed model, the optimal parameters help ANFIS have a more accurate estimation. At each iteration, the DE algorithm can generate various parameters such that the best model is used in the final model.

4. MODEL EVALUATION

4.1. Evaluation criteria

The model assessment criteria are presented in this section. As the assessment criteria play a main role in the proposed model, the criteria selection and determination can affect the success of the research because various assessment criteria are used in different papers and in most cases, the program source code is not available. Therefore, in this research attempts were made to use the most general assessment criteria utilized in a majority of papers. The estimation models' performance is investigated using several criteria, including RE. This is usually known as the first and main criterion and is shown in (2). Another criterion obtained using the previous equation is MRE that calculates the absolute error percentage compared to the real effort and is shown in (3). The mean of MREs is called MMRE and is shown in (4). Additionally, the median of MREs is called MdMRE.

$$RE = \frac{Estimated\ value - actual\ value}{actual\ value} \quad (2)$$

$$MRE_i = \frac{|Estimated\ value - actual\ value|}{actual\ value} \quad (3)$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE_i}{N} \quad (4)$$

The other important criterion is the performance index PRED(x), showing the percentage of predictions that correctly detect x value and is defined in (5).

$$PRED(x) = \sum_{i=1}^{N} D_i * \frac{100}{N} \quad (5)$$

$$D_i = \begin{cases} 1 & if\ MMRE < \frac{x}{100} \\ 0 & otherwise \end{cases}$$

$$when\ x = 25\ the\ PRED\ metric\ is\ defind\ as\ PRED(0.25)$$

Reducing MMRE and increasing PRED are the main goals of all software cost and effort estimation models. As they are among the most reliable and common criteria, they are also used in this research. In order to assess, Albrecht and Kemerer datasets were used for software efforts estimation because these two software measurement datasets have been used in the related works as well, and they are good criteria to measure the required method. Albrecht dataset includes developed projects using the third-generation language and contains five independent features; Filcount, SLOC, Quecount, Outcount, and Inpcout. Moreover, there are two dependent features called FP and Effort. The "Effort" feature that indicates the required effort is in terms of each 1000 persons per hour. Kemerer dataset includes the data gathered from 15 large commercial data processing projects. Each project in this dataset has seven input features, including duration, programming language, hardware, adjusted functional points (AdjFP), Effort, KSLOC, and RAWFP (raw functional points) [26].

4.2. Experimental design

To evaluate and assess the proposed method, first, the data should be pre-processed using various methods so that the existing data can be used in datasets in the best way. The first step in data pre-processing is standardization. One method is to convert data to a new set in which all values are between zero and one. This is also called normalization [27]. (6) shows normalization.

$$z_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \quad (6)$$

where z_i shows normalized data, xi is the initial value of data. Moreover, x_{min} and x_{max} show the minimum and maximum data, respectively.

Then data are divided into two datasets: training data and test data. To this purpose, 3-fold cross-validation is used. In this classification method, the normalized datasets are randomly divided into three groups with approximately equal sizes. In the first stage, two groups are considered for training (e.g., s2 and s3). The other group (e.g., s1) is then used for testing. Indeed, when datasets are divided into three datasets s1, s2, and s3, one of them (e.g., s1) is selected for testing, and the other two datasets (s2 and s3) are used for training separately. All possible states are shown in Table 1.

Table 1. How to apply 3-fold cross-validation on the proposed method

	S		
S1	S2	S3	S3
Possible sequences			
Base	Train	Test	Test
Fold 1	S1	S2	S3	S3
	S1	S3	S2	S2
Fold 2	S2	S1	S3	S3
	S2	S3	S1	S1
Fold 3	S3	S2	S1	S1
	S3	S1	S2	S2

The main reason for selecting k to be 3 (k=3) is to have the same conditions for making a comparison between the results of this study and those of the base research [25]. This continues until all datasets are used for testing. Then in the test phase, effort estimation is carried out. And at the last stage, MMRE and PRED (0.25) are calculated for each group. Finally, the mean of the three obtained accuracies is considered as the final result.

4.2.1. Initial setting of parameters

In this research, genfis function was used in MATLAB software, and a fuzzy basis system was implemented. To obtain the best possible result in this system, the default setting of parameters was changed, and the parameters were set with higher accuracy. Table 2 displays the values used for these variables. As ANFIS default training functions were not used in this research, the DE algorithm was used for setting parameters more accurately. The application of the DE algorithm has a significant effect on final results. The parametric values of the DE algorithm are shown in Table 3. The remarkable point in determining the values of this algorithm's parameters is that all of the values are obtained through a comprehensive trial and error process.

Table 2. Parameters value

Parameter	Value
Input MF function type	Gaussian
Output MF type	Linear
Based fuzzy system	Genfis
Learning algorithm	DE
And method	prod
Im method	prod
Or method	probor
Defuzz method	wtaver
Learning algorithm	DE
Agg method	sum

Table 3. DE parameters value

Parameter	Value
Max. no. of generation	200
npop	80
Scaling factor	0.8
Crossover rate	0.5

4.3. Comparing the proposed method with other models

In order to test the proposed method, it was compared with other methods. To ensure the accuracy and precision of the comparison, it should be considered that the quality criterion of assessment functions and selected intervals are the same for all data. In this research, the results of the proposed model are compared with ANFIS-SBO model, which is related to the relevant work. Additionally, the comparison was made using the other DE algorithm, i.e., genetic algorithm. Moreover, as the basis of the work in this research was studying neuro-fuzzy networks, the results of the proposed model were also compared with those of ANFIS network that uses artificial neural networks (gradient descent). Results obtained from Albrecht dataset are represented in Table 4.

Table 4. Results obtained from Albrecht dataset in comparison to other models

Models	Training Set		Testing Set	
	MMRE	PRED	MMRE	PRED
ANFIS-ANN	0.81	0.3	0.74	0.2
ANFIS-DE	0.3	0.78	0.28	0.86
ANFIS-SBO	0.5	0.47	0.51	0.42
ANFIS-GA	0.32	0.54	0.38	0.6

Also, the values obtained from MMRE and PRED (0.25) of the dataset are compared to other methods and shown in Figure 1. The results show that the proposed method produced more accurate estimation compared to other models and can significantly improve MMRE and PRED (0.25) values. The results of using the proposed model in Kemerer dataset in comparison to other popular models are shown

in Table 5. In Figure 2, the results obtained from MMRE and PRED (0.25) in Kemerer dataset are compared with other models. The proposed model revealed the best results compared to other models.

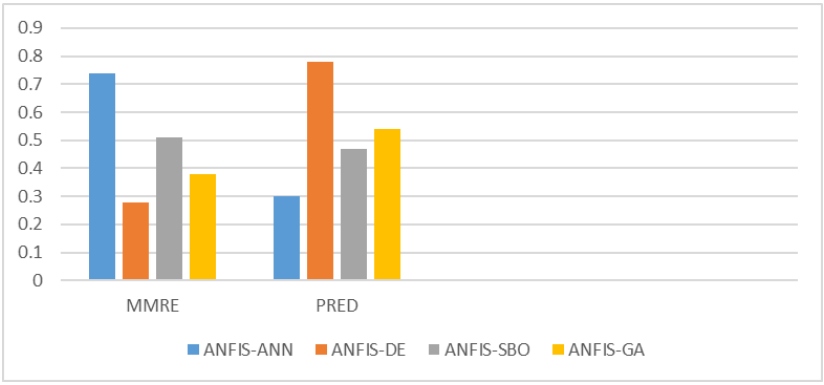

Figure 1. PRED and MMRE criteria values plot in Albrecht dataset

Table 5. Results obtained from Keremer dataset in comparison to other models

Models	Training Set		Testing Set	
	MMRE	PRED	MMRE	PRED
ANFIS-ANN	0.05	0.9	0.13	0.7
ANFIS-DE	0.77	0.21	0.9	0.31
ANFIS-SBO	0.07	0.89	0.08	0.9
ANFIS-GA	0.6	0.20	0.75	0.32

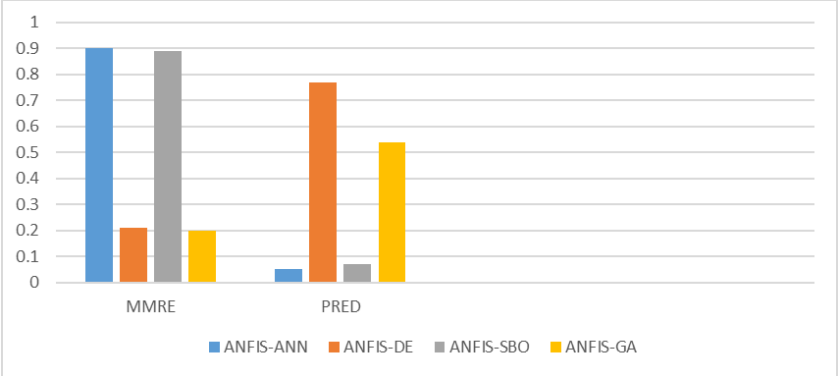

Figure 2. PRED and MMRE criteria values plot in Kemerer dataset

4.4. Improvement analysis

As shown in Tables 1-5, the MMRE criterion values are less than their similar values in other algorithms. This holds for both datasets used and is shown in Figures 1 and 2 more clearly. Additionally, this holds for the value of the PRED (0.25) criterion, and as shown in Tables, the value of this criterion is the highest value in both datasets. This criterion values are shown in Figures 1 and 2 for greater clarity. In Table 6, the percentages of improvement obtained by applying the proposed method on Albrecht dataset are shown with MMRE and PRED (0.25).

Table 6. The percentage of obtained improvement by applying the proposed method on Albrecht datasets

Models	MMRE	PRED
ANFIS-ANN	46%	48%
ANFIS-SBO	23%	31%
ANFIS-GA	10%	24%

As shown in Table 6, the highest improvement of MMRE is 46%, which is related to ANFIS basic method that uses artificial neural networks for training. This is while the lowest improvement percentage (10%) is related to the genetic algorithm (ANFIS-GA). Like MMRE, PRED (0.25) has improved as well. The lowest improvement percentage (24%) is related to S-GA, and the highest improvement percentage of PRED (0.25) (48%) has occurred for ANFIS-ANN. The improvement percentages obtained using the proposed method applied to Kemerer dataset are shown in Table 7 with MMRE and PRED (0.25) assessment criteria.

Table 7. The percentage of obtained improvement by applying the proposed method on Kemerer datasets

Models	MMRE	PRED
ANFIS-ANN	70%	72%
ANFIS-SBO	69%	7%
ANFIS-GA	1%	17%

As shown in Table 7, the proposed method can significantly improve PRED (0.25) metric compared to other methods. The highest MMRE improvement (70%) is related to ANFIS-ANN. Moreover, considering the greatest improvement, ANFIS-SBO comes second after ANFIS-SBO with a relatively similar percentage, i.e., 69%. This is while the lowest percentage, which is only 1%, is related to ANFIS-GA.

In the other assessment criterion, the lowest improvement for PRED (0.25) is related to ANFIS-SBO model, i.e., 7%, while the greatest improvement of 72% has to do with the basic model, i.e., ANFIS-ANN. It is obvious that the improved percentage for PRED (0.25) is higher than MMRE, but the proposed method is still stronger than other models. Finally, by investigating the obtained results, it was found that the proposed model in both datasets revealed more accurate results compared to other models.

5. CONCLUSION

Presenting a reliable and accurate software effort estimation is a significant issue for software project teams and companies. Considering the uncertainty existing in software projects, not only should a method be selected to resolve this uncertainty, but also a method should be offered for accurate software development effort estimation. ANFIS can investigate the uncertainty of software projects by correctly selecting membership functions and training algorithms. This study introduced the proposed method called ANFIS-DE and used it for a more accurate estimation of software projects. Though ANFIS is a popular estimation model widely used in software efforts estimation, this model is still incapable of generating estimations with proper accuracy in most cases. As a solution for this problem, evolutionary algorithms, especially the DE algorithm, can be used as an alternative to class training algorithms in ANDIS. This research focused on using hybrid ANFIS-DE as a new training algorithm. The intrinsic features of DE made this algorithm appropriate about features weighing, weight adjustment, deviation, and ANFIS parameters adjustment. Additionally, this algorithm is among the evolutionary algorithms with the highest convergence rate. The assessment criteria to test and compare ANFIS-DE with other known algorithms, including GA, SBO, and neuro-fuzzy system-based ASS, were completely investigated and used. The results from this research showed that the proposed method has been able to obtain better performance in all of the assessed criteria than other algorithms in experimental functions.

REFERENCES

[1] R. S. Pressman, “Software Engineering: A Practitioner's Approach,” 7th edition ed. *New York: McGraw-Hill Science/Engineering/Math*, 2009.
[2] S. McConnell, “Rapid development: taming wild software schedules,” *Microsoft Press, 1 edition*, 1996.
[3] A. Idri and I. Abnane, "Fuzzy analogy based effort estimation: An empirical comparative study," in *2017 IEEE International Conference on Computer and Information Technology (CIT)*, Helsinki, pp. 114-121, 2017.
[4] A. Idri, M. Hosni, A. Abran, "Systematic literature review of ensemble effort estimation," *Journal of System and Software,* vol. 118, pp. 151-175, 2016.
[5] A. Zaid, M. H. Selamat, A. A. A. Ghani, R. Atan, and K. Wei, "Issues in software cost estimation," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 11, pp. 350-356, 2008.
[6] F. A. Amazal, A. Idri, and A. Abran, "Improving fuzzy analogy based software development effort estimation," in *2014 21st Asia-Pacific Software Engineering Conference*, Jeju, pp. 247-254, 2014.
[7] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on software engineering*, vol. 33, no. 1, pp. 33-53, 2007.

[8] T. J. Gandomani, H. Faraji, and M. Radnejad, "Planning Poker in cost estimation in Agile methods: Averaging vs. Consensus," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, Tehran, Iran, pp. 066-071, 2019.
[9] S. G. MacDonell, M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management," *Journal of Systems and Software,* vol. 66, no. 2, pp. 91-98, 2003.
[10] N. Mittas and L. Angelis, "Combining regression and estimation by analogy in a semi-parametric model for software cost estimation," in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pp. 70-79, 2008.
[11] U. Subbiah, M. Ramachandran, and Z. Mahmood, "Software Engineering Framework for Software Defect Management Using Machine Learning Techniques with Azure," in *Software Engineering in the Era of Cloud Computing: Springer*, pp. 155-183, 2020.
[12] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A PSO-based model to increase the accuracy of software development effort estimation," *Software Quality Journal*, vol. 21, no. 3, pp. 501-526, 2013.
[13] P. Kumari and I. Singla, "Tuning of use case point (UCP) analysis Parameter using PSO," *Communications on Applied Electronics (CAE),* vol. 1, no. 5, pp. 25-28, 2015.
[14] H. Hota, R. Shukla, and S. Singhai, "Predicting Software Development Effort Using Tuned Artificial Neural Network," in *Computational Intelligence in Data Mining,* vol. 3, pp. 195-203, 2014.
[15] P. Rijwani and S. Jain, "Enhanced software effort estimation using multi layered feed forward artificial neural network technique," *Procedia Computer Science,* vol. 89, pp. 307-312, 2016.
[16] S. M. Satapathy, B. P. Acharya, and S. K. Rath, "Early stage software effort estimation using random forest technique based on use case points," *IET Software,* vol. 10, no. 1, pp. 10-17, 2016.
[17] A. B. Nassif, M. Azzeh, L. F. Capretz, D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Computing and Applications,* vol. 27, no. 8, pp. 2369-2381, 2016.
[18] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," *Swarm and Evolutionary Computation*, vol. 38, pp. 158-172, 2018.
[19] V. Sharma and H. K. Verma, "Optimized fuzzy logic based framework for effort estimation in software development," *International Journal of Computer Science Issues (IJCSI),* vol. 7, no. 2, pp. 30-38, 2010.
[20] C. Lopez-Martin, "A fuzzy logic model for predicting the development effort of short scale programs based upon two independent variables,"*Applied soft Computing,* vol. 11, no. 1, pp. 724-732, 2011.
[21] S. Kamal, et al., "A fuzzy logic based software cost estimation model," *International Journal of Software Engineering and Its Applications,* vol. 7, no. 2, pp. 7-18, 2013.
[22] S. Kad and V. Chopra, "Fuzzy logic based framework for software development effort estimation," *International Journal of Engineering Sciences,* vol. 1, pp. 330-342, 2011.
[23] V. Agrawal and V. Shrivastava, "Performance evaluation of software development effort estimation using neuro-fuzzy model," vol. 4, pp. 193-199, 2015.
[24] S. Nanda and B. Soewito, "Modeling software effort estimation using hybrid PSO-ANFIS," in *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, Lombok, pp. 219-224, 2016.
[25] S. H. S. Moosavi and V. K. Bardsiri, "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation," *Engineering Applications of Artificial Intelligence*, vol. 60, pp. 1-15, 2017.
[26] C. F. Kemerer, "An empirical validation of software cost estimation models," *Communications of the ACM*, vol. 30, no. 5, pp. 416-429, 1987.
[27] E. Kocaguneli, T. Menzies, "Software effort models should be assessed via leave-one-out validation," *Journal of Systems and Software,* vol. 86, no. 7, pp. 1879-1890, 2013.

BIOGRAPHIES OF AUTHORS

Amir Karimi received his Master's degree in software engineering from the Isfahan branch, Islamic Azad University, Iran in 2019. He is currently working in a software engineering reseach group. His research interests are software cost estimation, software project management, and software planning.

Taghi Javdani Gandomani received his Ph.D. degree from the University of Putra Malaysia (UPM), Malaysia in 2014. He is currently an assistant professor at Shahrekord University, Iran. His research interests are software development process, Agile software development, software cost estimation, and empirical software engineering.