Computing and Informatics, Vol. 39, 2020, 1148–1171, doi: 10.31577/cai_2020_6_1148

OPEN HYBRID MODEL: A NEW ENSEMBLE MODEL FOR SOFTWARE DEVELOPMENT COST ESTIMATION

Amin Moradbeiky, Vahid Khatibi Bardsiri

Department of Computer Engineering, Faculty of Science
Kerman Branch, Islamic Azad University
Kerman, Iran
e-mail: ampayam@yahoo.com, kvahid2@live.utm.my

Mehdi Jafari

Department of Electrical Engineering, Faculty of Engineering
Kerman Branch, Islamic Azad University
Kerman, Iran
e-mail: mjafari@iauk.ac.ir

Abstract. Given various features of a software project, it may face different administrative challenges requiring right decisions by software project managers. A major challenge is to estimate software development cost for which different methods have been proposed by many researchers. According to the literature, the capability of a proposed model or method is demonstrated in a specific set of software projects. Hence, the aim of this study is to present a model to take advantage of the capabilities of various software development cost estimation models and methods simultaneously. For this purpose, a new model called "open hybrid model" was proposed based on the firefly algorithm. The proposed model includes an extensible bank of estimation methods. The model also includes an extensible bank of rules to describe the relation between existing methods. Considering project conditions, the proposed model tries to find the best rule for combining estimation methods in the methods bank. Three datasets of real projects were used to evaluate the precision of the proposed model, and the results were compared with those of other 11 methods. The results were compared based on performance parmeters widely used to show the accuracy and stability of estimation models. According to the results, the open hybrid model was able to select the most appropriate methods present in the methods bank.

Keywords: Development cost estimation, firefly algorithm, software project

1 INTRODUCTION

According to the literature [30, 36, 5], various methods for software cost estimation can be classified as algorithmic and non-algorithmic strategies. Both strategies can be useful depending on the type of software projects. The efficiency will be increased by proper identification of the requirements of each method. Each method has its own advantages and disadvantages.

Moreover, different approaches have been recently used for software development cost estimation. For instance, back propagation learning algorithms on a multilayer perceptron and the genetic programming (GP) have been proposed for this purpose [24]. Reddy et al. [33] used the multi-objective particle swarm optimization (MOPSO) to develop a software cost estimation model which outperformed the standard constructive cost model (COCOMO). Andreou and Papatheocharous [2] employed fuzzy decision trees for software cost estimation.

A set of cost estimation methods are classified as algorithmic strategies. These methods employ mathematical models for estimating project costs. These models are defined as a function of cost factors.

Several algorithmic methods such as linear and nonlinear models, Putman's model, Seer-Sem model, function point model, Bailey and Basili model, Aron model, Doty model, COCOMO (constructive cost model), and COCOMO II have been proposed so far. COCOMO has been used by many researchers and thus is further discussed in this section.

COCOMO was first proposed by Boehm in 1981 as an empirical model by collecting data of different real-world software projects. The collected data are then analyzed to obtain certain formulas matching observations. In addition, Boehm et al. analyzed the developed version of COCOMO with more capabilities than the earlier version and released it as COCOMO II [7, 8, 9].

As mentioned earlier, algorithmic methods based on one or more mathematical formulas lack sufficient flexibility. Consequently, non-algorithmic models have been proposed for cost estimation.

Unlike algorithmic methods, non-algorithmic methods are based on analytical comparison and inference. The use of non-algorithmic methods requires accurate information on prior projects. Non-algorithmic methods make estimations by analyzing prior datasets without employing any specific relation or equation. The most common methods for estimating software criteria in non-algorithmic methods include exhaustive search, comparison, trial, error and inference. Some studies in this field are reviewed below:

Cuadrado-Gallego et al. [10] analyzed and compared machine learning and expert judgment methods that have been extensively used for cost estimation of software projects. To this end, they compared and pointed out advantages and disadvantages of seven machine learning methods including neural networks, fuzzy logic, comparative analysis, decision trees, case-based reasoning (CBR) and rule-based reasoning and hybrid systems. According to their results, CBR outperformed other

methods. In fact, this method could be successful in the absence of statistical relationships.

Wen et al. [41] systematically analyzed machine learning models for software cost estimation from four perspectives. They analyzed different studies completed between 1991 and 2010 in terms of machine learning, estimation precision, comparison models and estimation field. Different machine learning models have their own advantages and disadvantages; thus, they can be employed in different fields. Huang et al. [17] analyzed preprocessing as a major step of machine learning in software project cost estimation. They aimed at empirical evaluation of the effect of data preprocessing on machine learning methods for software cost estimation. Bardsiri and Hashemi [6] employed the correlation analysis approach to evaluate the efficiency and precision of five major machine learning methods for software project cost estimation. They also analyzed the effect of feature selection on the estimation precision.

Idri et al. [18] classified the papers published on CBR cost estimation by IEE, ACM, ScienceDirect, and Google Scholar from 1990 to 2012 in terms of methodology, type and technique. According to their findings, ABE methods outperformed other techniques, especially when combined with the fuzzy logic or genetic algorithm. Sigweni and Shepperd [37] systematically reviewed and evaluated feature weighting techniques in the ABE method for software development cost estimation. They analyzed different aspects of each technique in addition to advantages, disadvantages and efficiency on different datasets. For this purpose, they comprehensively reviewed all papers in this area published from 2000 to 2014.

González-Ladrón-de-Guevara et al. [15] analyzed software development cost estimation in the ISBSG dataset. They reviewed different studies on the ISBSG dataset (2000–2013) to determine those ISBSG variables used in estimations and to examine the effect of ISBSG variables on the development of cost estimation models. They also determined dependent or independent variables. Out of 71 ISBSG variables, 20 variables were used as independent variables in most studies. Their findings can help other researchers in selecting appropriate variables in cost estimation models. General algorithms can be used to develop a model from databases. The EM clustering algorithm [14] was combined with the database segmentation method leading to an exact model and estimate for size-effort criterion and standard quality criterion. The estimation criterion improved the accuracy of expected parameters in each segment using EM clustering algorithm and local regression.

The use case points-activity based costing (UCPabc) method and function points (FP) can be employed for software development cost estimation. Azzeh and Nassif used UCPabc for software development cost estimation [4]; however, Dewi et al. used FP for this purpose [12]. Dewi et al. compared FP and UCPabc and found that the latter was much more accurate than the former [11].

Pospieszny et al. [31], Mensah et al. [26], Puspaningrum and Sarno [32], Moosavi and Bardsiri [27], Wani and Quadri [40], Arora and Mishra [3], Abnane et al. [1],

and Khuat and Le [20] respectively employed a linear method integrated with neural networks, multivariate regression model, artificial neural networks integrated with the harmony algorithm, neural networks integrated with the fuzzy method, neural networks, the fuzzy model and the artificial bee colony algorithm for software development cost estimation.

Rastogi et al. [34] evaluated software cost estimation techniques and models extensively. They compared and classiffied different methods and considered cost estimation techniques resulting in a higher precision as a selection criterion. According to their results, all cost estimation techniques have their own advantages and disadvantages. They also believe that there is no single method which can be accepted by all researchers. Therefore, a combination of different methods should be employed to achieve realistic cost estimation. Shekhar and Kumar [36] reviewed and analyzed different software cost estimation techniques and models. For this purpose, they analyzed advantages and disadvantages of different methods and concluded that no single method could be used as the best cost estimation method and a more accurate estimate could be achieved by combining different methods. Pandey [30] classiffied software project cost estimation methods into parametric and nonparametric models by analyzing different cost estimation techniques and addressing their advantages and disadvantages. Khatibi and Jawawi [19] reviewed and analyzed different software cost estimation methods from different points of view. Each of the cost estimation methods can be employed efficiently in certain projects and situations. The performance of any estimation method depends on different parameters such as project complexity, project span, etc.

Ensemble models have been presented for estimating software development cost by another group of scholars. In these models, independent methods attempt to predict software development cost separately. The estimates from different methods are then combined with a method to calculate the final estimate. Various estimators and combinators have been used in the literature. Studies by Wu et al. [42], Kocaguneli et al. [22], Elish [13], and Hsu et al. [16] can be noted in this regard.

The reviewed studies suggest a single method or model can offer high estimation accuracy only in a limited number of datasets. Ensemble models allow benefiting from the advantages of different models and methods at the same time. A review of ensemble models reveals they are based on a limited number of estimators and combiners. Further, a survey of previous ensemble models shows using the same combiner fails to produce the most accurate estimations in all datasets. This study proposes an ensemble model that is not dependent on a set number of estimators and combiners, and enabling it to combine its estimators intelligently and by the best approach.

This article is organized as follows: Section 2 describes the background and related work. Section 3 describes the firefly algorithm. Different criteria are introduced to evaluate the precision of the model proposed in Section 4. The proposed model and its evaluation method are introduced in Section 5 and Section 6, respectively. The datasets of real projects introduced in Section 7 are used for testing. The

results of testing the proposed model are compared to those of other 11 methods to determine the superiority of the proposed model. These 11 methods are presented in Section 8. Section 9 presents the results of testing the proposed model. The results are analyzed statistically in Section 10. Conclusions and future work are reported in Section 11.

2 BACKGROUND AND RELATED WORK

Ensemble models can be used in various data mining fields. Malgonde and Chari [25] used an ensemble model for estimating agile software development cost. Silva et al. [38] combined data mining techniques for predicting the export potential of a company. In an article on network security, Ochieng et al. [28] described identification of worms by an ensemble model. Salehi et al. [35] proposed an ensemble model by data mining techniques for cancer detection.

Various ensemble models have also been provided for software development cost estimation. According to the literature on software development cost prediction methods, ensemble models which include multiple predictors as a peer prediction model give more accurate results than each of the individual methods [22]. Ensemble models rely on multiple methods so that the inability of a method in providing an accurate estimate can be compensated by the accurate estimates provided by other methods and this is the main reason behind the success of these models [22].

Figure 1 shows the architecture of ensemble models. In general, ensemble models consist of two important parts. The first part includes a set of individual estimation models or methods ($f_{1...m}$). The second part includes a set of various methods for combining the estimations ($C_{1...k}$) obtained from estimators in the first part. Each combinator calculates an independent estimation. Accordingly, one can conclude that if Part 1 or 2 or both include a diverse and accurate set of various models or methods, the final model will be able to provide exact estimates in different datasets.

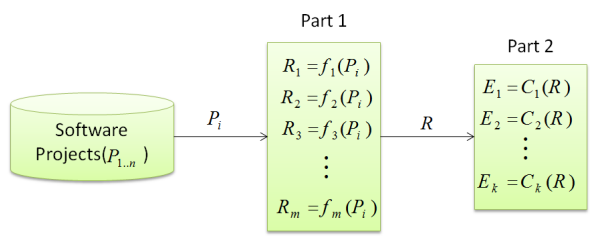


Figure 1. Architecture of ensemble models

Wu et al. [42] proposed an ensemble model for estimation of software development cost. For this purpose, they combined estimates obtained from different CBR methods by four linear combination techniques, namely median combination, mean combination, weighted mean combination (WMC) and outperformance combination (OC). The proposed model in their study was tested on two datasets. According to

the results of these two datasets and based on the evaluation criterion MMRE, the WMC combination techniques provided more accurate results in one of the datasets, whereas the other combination technique (mean) led to more accurate results in the other dataset. The results of tests based on the evaluation criterion MdMRE also confirmed this point.

Elish [13] proposed an ensemble model for software development cost estimation. He combined the estimates obtained from three different methods by eight combination methods. The model proposed by Elish was tested by different datasets. According to the results, different combination methods led to best results based on the same evaluation criterion used in various datasets.

Hsu et al. [16] developed an ensemble model consisting of COCOMO, linear regression, CBR, grey relational analysis and artificial neural networks estimators. Equally weighted combination, median weighted combination and weighted adjustment based on a criterion were used for combining the estimations. The final model was tested by different datasets. The results obtained from different combination methods in various datasets indicated that no certain combination method was able to obtain the best results in all datasets.

Song et al. [39] provided an ensemble model of five machine learning methods for estimating software development cost. Kultur et al. [23] proposed an ensemble model using neural networks. Pahariya et al. [29] provided an ensemble model in which the results from computational intelligence techniques are combined by three different methods to calculate the final estimate.

According to the literature, no certain combination method was able to calculate the most accurate estimation in the different datasets, which raises the question whether it is possible to provide an intelligent model to detect the ability of different estimation methods in various datasets and select the best combination method.

The accuracy of ensemble estimation models is dependent on the accuracy and diversity of models and methods in the 1 and 2 parts (Figure 1, the architecture of ensemble models). According to the literature, the ensemble model is dependent on a certain and limited set of estimator and combinator methods. On the other hand, there are very diverse estimation and combination methods and new methods are still added to this set. The important question raised here is whether a proposed hybrid model can generate accurate results regardless of accuracy obtained from combination or estimation methods used in its parts of 1 and 2 (Figure 1, the architecture of ensemble models). Simply speaking, is it possible to eliminate factors limiting the accuracy of the final model due to limitations of models or methods used in the 1 or 2 part (Figure 1)? The model proposed in this article is able to eliminate the above-mentioned limitations.

3 FIREFLY ALGORITHM

Firefly algorithm (FA) was first introduced by Xin-She Yang in 2009 [43]. FA is a meta-heuristic optimization algorithm that imitates the social behavior of fireflies

flying in the tropical and temperate summer sky. FA algorithm has been used based on three principles:

1. Each firefly is capable of attracting other fireflies.
2. Attraction of each firefly depends on the level of its light so that the one with more light attracts a firefly with less light. If no firefly has more light, one is selected randomly.
3. The light of each firefly is under the influence of its distance from the goal.

As illustrated in Figure 2, each firefly generates a random solution. Then, some parameters as light intensity, primary attractiveness, and absorption coefficient are defined. Then, the most brilliant firefly is selected. Fireflies move toward a more brilliant firefly. Moving toward each other, fireflies light decreases and their attractiveness varies. Next, the best firefly is picked up for repetitive cycle according to an objective function. This process continues until the end condition is done.

The firefly attractiveness is due to its light determined by the objective function from the problem. As the most optimized method, the I light of firefly in the specific location of X can be selected as $I(x) \propto f(x)$. β is the relative attractiveness seen by each firefly. Therefore, it would change with r_{ij} distance between i firefly and j firefly. Additionally, light intensity decreases taking distance from the source. Light is absorbed in media. Therefore, attractiveness varies as its ratio varies.

```
Objective function f(x), x = (x1... xd)T
Generate initial population of fireflies xi (i = 1, 2... n)
Light intensity Ii at xi is determined by f (xi)
Define light absorption coefficient γ
while (t < MaxGeneration)
   for i = 1 : n all n fireflies
      for j = 1 : n all n fireflies (inner loop)
         if (Ii < Ij),
            Move firefly i towards j;
         end if
         Vary attractiveness with distance r via exp[−γ r]
         Evaluate new solutions and update light intensity
      end for j
   end for i
   Rank the fireflies and find the current global best g*
end while
```

Figure 2. Pseudocode of firefly algorithm [43]

Each firefly owns its specific attractiveness

$$\beta(r) = \beta_0 e^{-\gamma r^m}, m \geq 1. \tag{1}$$

The distance between two fireflies of i and j is calculated through following formula

$$r_{ij} = |x_i - x_j| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2}. \tag{2}$$

The movement of fireflies attracting more fireflies is calculated through the following formula

$$x_i = x_i + \beta_0 e^{\gamma r_{ij}^2}(x_j - x_i) + \alpha\left(rand - \frac{1}{2}\right). \tag{3}$$

Five parameters in the FA should be set for optimization purposes generally called FA configuration for problem solving. The settings of these parameters vary with the problem under study and expected objectives of the algorithm [43]. These settings determine the behavior of the algorithm during problem solving. The FA parameters are as follows:

1. N represents the number of fireflies used for problem solving.
2. *MaxGeneration* indicates the number of iterations.
3. *Alpha* (α) is a coefficient within the range $[0, 1]$ and is multiplied by the random number.
4. *Betamin* (β min) represents the minimum Beta (β) value, which is indicative of attractiveness of the light source.
5. *Gamma* (γ) is determined considering attractiveness variations. This parameter plays a key role in convergence rate and the behavior of FA.

A hyper-parameters tuning step was used for precise FA conguration. In this step, the best α and γ were obtained in the $[0, 1]$ and $[0.1, 20]$ ranges, respectively, by a comprehensive trial and error process.

4 ESTIMATION ERROR DETERMINATION EQUATIONS

Specific metrics were employed in this study to calculate the estimation error. These metrics have been used in several studies to compare the research results with those of other similar studies. The metrics include relative error (RE), magnitude of relative error (MRE), mean magnitude of relative error (MMRE), median magnitude of relative error (MdMRE) and prediction percentage (Pred), shown by metrics (4),

(5), (6), (7) and (8):

$$\mathrm{RE} = \frac{\mathrm{Estimate} - \mathrm{Actual}}{\mathrm{Actual}}, \tag{4}$$

$$\mathrm{MRE} = \frac{|\mathrm{Estimate} - \mathrm{Actual}|}{\mathrm{Actual}}, \tag{5}$$

$$\mathrm{MMRE} = \mathrm{Mean}(\mathrm{MRE}), \tag{6}$$

$$\mathrm{MdMRE} = \mathrm{Median}(\mathrm{MRE}), \tag{7}$$

$$\mathrm{PRED}(X) = \frac{A}{N}. \tag{8}$$

5 PROPOSED MODEL

Different methods have been proposed for estimating important project parameters such as cost. According to the literature, the methods can only operate properly in limited datasets because of dependence on a certain estimation technique. According to this point, the challenge addressed in this study was to determine in which datasets a method would operate more successfully, to what extent it would be successful and how it would be possible to employ its high precision. For this purpose, a new model called "open hybrid model" was proposed based on the FA. It is also possible to add different estimation methods to the methods bank of the proposed model.

5.1 The Open Hybrid Model

The proposed model in this study includes a bank of various estimation methods. The main idea of the model is that an estimator machine, based on a specific method such as the ABE, is capable of proper cost estimation only in few datasets. Accordingly, the first objective of the proposed model is to test the precision levels of different methods for estimating a set of projects. Then it intends to allocate a specific value to each method with respect to its precision. This value indicates the effectiveness of a method on the final estimation. The model first divides the datasets of projects into basic, training and testing projects. The proposed model operates in two stages: training and testing. In the training stage, the open hybrid model operates on basic and training projects. The training stage aims at evaluating the precision of each method existing in the methods bank and also achieving the best configuration for the optimal use of each method. In the testing stage, the open hybrid model operates on the basic and testing projects to evaluate the precision of configuration obtained from the training stage.

5.2 Training

Figure 3 shows the training flowchart. As mentioned earlier, the training stage operates on the basic and training projects. There are two banks of methods and rules in this stage. The methods bank consists of different methods for software development cost estimation. The rules bank includes different rules for integration of estimation methods in the methods bank. The coordinator function (COF) in the training stage employs the methods bank and rules bank. The COF needs a set of projects for estimation. It makes use of other projects resembling the one which should be estimated (P') to increase the estimation precision. To find projects similar to P', it is clustered along with the basic projects based on specific features (f_list). The f_list is a set of projects features recommended by FA. Based on the accuracy of estimation from the training set, the FA attempts to recommend a better f_list in each iteration, selecting more similar projects to P' for its estimation. When proposing f_list, FA is only allowed to recommend from continuous features. The P'-containing cluster is then sent to the COF, which uses the received projects to estimate P' based on the rule (proposed by FA) taken from the rules bank (this specific rule determines how to use existing methods in the methods bank). Table 1 shows the rules in the rules bank where ES_i represents the value of current project estimated by i^{th} method, and W_i (proposed by FA) indicates the coefficient of the i^{th} method existing in the methods bank. In some rules, er and m have been used to reduce the estimation error. This study used the k-means clustering method in which the number of clusters, denoted by k, should be adjusted. This parameter is recommended by FA during the training stage and then improved in the later iterations. The obtained k value is then used for testing.

#	function
Rule 1	$Cost = W_1 \times ES_1 + W_2 \times ES_2 + \cdots + W_n \times ES_n$
	$Cost = Cost + Cost \times m$
	$Cost = Cost + er$
Rule 2	$Cost = Median(ES_{1\ldots n})$
	$Cost = Cost + Cost \times m$
	$Cost = Cost + er$
Rule 3	$Cost = Mean(ES_{1\ldots n})$
	$Cost = Cost + Cost \times m$
	$Cost = Cost + er$
Rule 4	$Cost = ES_i$ (ES_i is best method in training iteration)

Table 1. Rules bank

In each stage of the open hybrid model, the FA proposes k and f_list for clustering the projects and a rule with the required parameters and coefficients (W, er, m) of that rule to the COF in each iteration. The COF uses the proposed k, f_list and rule to estimate each and every project of the training set. Finally, the estimation error of the training set is determined and returned as the feedback of

the proposed k, f_list and rule to the FA. In the next iteration, the FA tries to propose k, f_list and rule with more appropriate parameters and coefficients to the COF. The output of the training stage includes the best k, f_list and rule along with the proposed parameters and coefficients required by that rule.

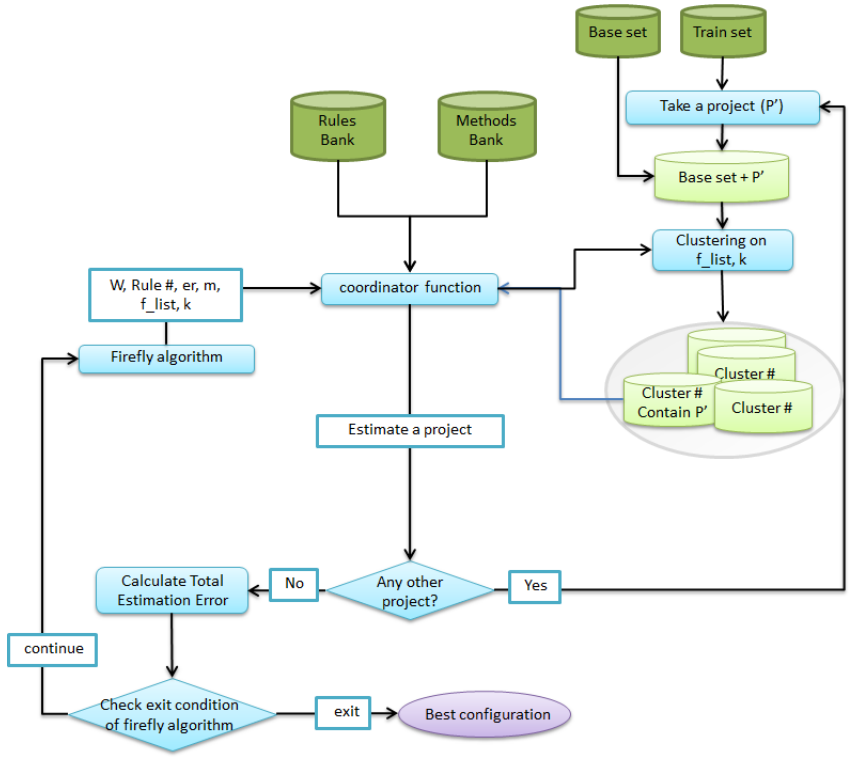


Figure 3. The training stage of the open hybrid model

5.3 Testing

As mentioned above, the testing stage operates on the basic and testing projects. According to Figure 4, the resulting configuration of the previous stage is given as the input to the model. The open hybrid model selects a project from the testing set (P') and then clusters P' with basic projects based on the f_list and k obtained from the previous stage. The cluster including P' is then given to the COF for estimating the development cost of P' based on the rule obtained from the previous stage. Another project is then selected and estimated through the previous stages. Iterations continue until no other projects remain in the testing set. Finally, the estimation error of each project is used to determine MdMRE, MMRE, and Pred. The results were used for comparing the models.

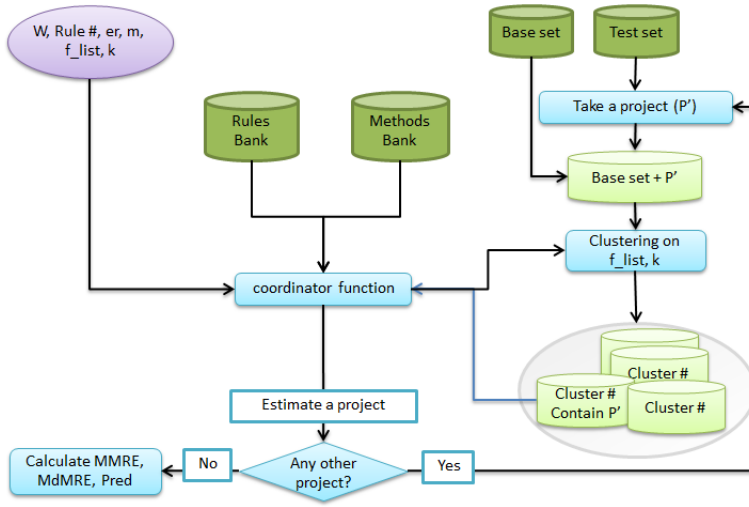


Figure 4. The testing stage of the open hybrid model

As the notable advantage of the open hybrid model over previous methods, it has been designed to add new methods and rules to the methods and rules banks. The model is also able to measure the efficiency of the new method intelligently and use it proportionately to its efficiency. Considering the methods used in the methods bank, the existing methods should be so diverse that model can operate dynamically in different modes and adapt to new conditions. The model proposed in this paper was obtained by analyzing the results of numerous studies indicating the success of any method in some datasets.

6 EVALUATION METHOD

The arrangement of samples in the training or testing stage may significantly affect the results of estimation models [21]. Therefore, a method is required to show the independency of results from the arrangement of samples to demonstrate the stability of results produced by the proposed model. Different evaluation methods such as 3-fold, 10-fold, and leave one out (LOO) have been proposed for this purpose. The LOO method was utilized in this study. In this method, when all projects except for one are used in the training stage, only one project is used in the testing stage. The nested LOO cross-validation was adopted in the compared methods. During training, different possible scenarios were considered for adjustable parameters by the LOO method, and the best quantities were used for testing. The process continued until tests completed for all projects.

7 DATASETS

The Desharnaise dataset contains data of 81 software projects collected from Canadian software houses. The software projects in this dataset have been described by 11 features. The dependent cost feature is based on 1000 person-hours. Ten independent features in this dataset include 'TeamExp', 'ManagerExp', 'YearEnd', 'Duration', 'Transactions', 'Entities', 'AdjFP', 'AdjFactor', 'RawFP', and 'Dev.Env'. Given that the data of 4 projects out of 81 projects in this dataset is missing and not available, the tests were conducted using the remaining 77 projects.

The Albrecht dataset contains data on software projects designed by third generation programming languages (3GLs). This dataset contains information on 24 projects. There is a dependent feature called 'work hours' in this dataset based on 1 000 hr. There are also 7 independent features ('input count', 'output count', 'query count', 'file count', 'line of code', 'RawFP', and 'function points') in this dataset.

The Kemerer dataset contains data on 15 software projects. The software projects in this dataset have been explained by 6 independent features and 1 dependent feature. The independent features include 'Language', 'hardware', 'RawFp', 'Duration', 'KSLOC' and 'AdjFP'. 'Cost' is considered as a dependent feature in this dataset and is measured by man-months. Table 2 shows the specifications of these datasets.

#	Dataset	Number of Projects	Number of Features	Mean of Cost
1	kemerer	15	7	219
2	desharnise	77	11	4 833
3	albrecht	24	8	21

Table 2. Datasets

8 TECHNIQUES

The results obtained from testing the proposed model were compared with those of different models to evaluate its precision. The following models were used for comparison:

Voting Ensemble: Elish [13] proposed an ensemble model for software development cost estimation. He combined the estimates obtained from three different methods by eight combination methods

Linearly Weighted Combinations Model (LWCM): Hsu et al. [16] developed an ensemble model consisting of COCOMO, linear regression, CBR, grey relational analysis and artificial neural networks estimators. Equally weighted combination, median weighted combination and weighted adjustment based on a criterion were used for combining the estimations.

Multilayer Perceptron (MLP): The neural network is a nonlinear modeling technique and MLP is a widely used neural network based on a network of neurons on an input layer with one or more hidden layers and an output layer.

Analog Based Estimation (ABE): ABE searches for the most similar sample to that which should be estimated. This method employs its internal functions to determine the resemblance of samples. The number of similar samples used for estimation is determined by the parameter K.

PSO + ABE: An estimation model based on ABE and particle swarm optimization (PSO) algorithm to increase the accuracy of software development cost estimation.

Ordinary Least Squares (OLS): OLS, as a regression-based method, is employed to determine the best regression line by minimizing the total squares.

Robust Regression (RoR): RoR is a regression-based method which can operate more accurately by weighting against unconventional data.

Multivariate Adaptive Regression Splines (MARS): MARS is a nonlinear nonparametric regression method with some interesting features such as ease of interpretability, modeling nonlinear complicated relationships and quick model development.

Classification and Regression Trees (CART): CART is an algorithm in which decision trees are employed for classification.

M5: This method can be regarded as a new type of CART. The model tree created by this method considers a linear regression for each leaf instead of determining a single value.

Least Squares SVM (LSSVM): The support vector machine (SVM) is a nonlinear machine learning method capable of mapping the input state space onto a space of higher dimensions leading to development of simpler linear regression functions.

9 TESTING THE OPEN HYBRID MODEL

Albrecht, Desharnise and Kemerer datasetes were employed to test the open hybrid model. The methods bank of the proposed model included CART, SWR, MLR, ABE and LSSVM. No normalization processes were performed on the data, and all of the methods were treated similarly and equally. The results of testing the proposed model on each dataset are described below.

Table 3 compares the results obtained from testing the open hybrid model on Albrecht with other methods. The proposed model was compared with other methods in terms of three different criteria (MMRE, MdMRE, and Pred). Accordingly, the proposed model was 51 %, and 65 % more precise than the most precise method in MMRE (Voting Ensemble), and the most precise method in MdMRE (LWCM), respectively. These results also indicated the higher precision of the proposed model

than other methods. Figure 5 shows the results on a diagram for a better comparison. As clearly seen, the proposed model is able to increase the estimation precision simultaneously in terms of MdMRE, MMRE, and Pred.

Method	MMRE	MdMRE	Pred
Open Hybrid	0.22	0.08	0.62
LWCM	0.63	0.23	0.62
Voting Ensemble	0.45	0.37	0.41
ABE	0.85	0.38	0.29
CART	1.04	0.51	0.33
LSSVM	0.88	0.63	0.33
M5'	0.89	0.46	0.25
MARS	0.87	0.29	0.45
MLP	0.95	0.41	0.2
OLS	0.79	0.5	0.37
PSO + ABE	1.04	0.48	0.12
ROR	0.72	0.66	0.29

Table 3. The results of testing Albrecht in comparison with other methods

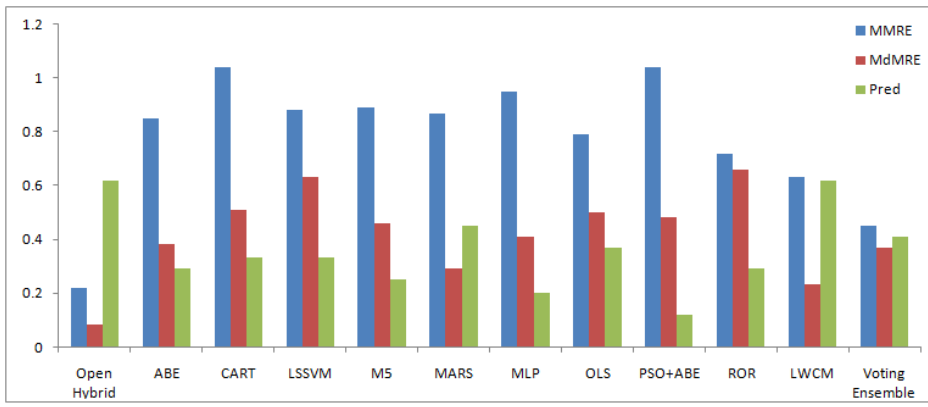


Figure 5. Comparing different methods in MMRE, MdMRE, and Pred on Albrecht

Table 4 compares the results of testing the open hybrid model on Kemerer with other methods. The results were evaluated in terms of MMRE, MdMRE, and Pred. Accordingly, the proposed model was nearly 54 %, 67 %, and 43 % more precise than the most precise method in MMRE (LWCM), the most precise method in MdMRE (LWCM), and the most precise method in Pred (Voting Ensemble), respectively. For a better comparison, the test results were shown on a diagram in Figure 6 which clearly depicts the precision of the proposed model on Kemerer in comparison with other methods.

Method	MMRE	MdMRE	Pred
Open Hybrid	0.23	0.09	0.66
LWCM	0.5	0.28	0.33
Voting Ensemble	0.54	0.36	0.46
ABE	0.82	0.46	0.2
CART	0.96	0.61	0.06
LSSVM	0.64	0.55	0.26
M5'	1.15	0.73	0.2
MARS	1.4	0.8	0.26
MLP	0.57	0.49	0.33
OLS	0.64	0.5	0.26
PSO + ABE	0.61	0.47	0.33
ROR	0.62	0.36	0.13

Table 4. The results of testing Kemerer in comparison with other methods

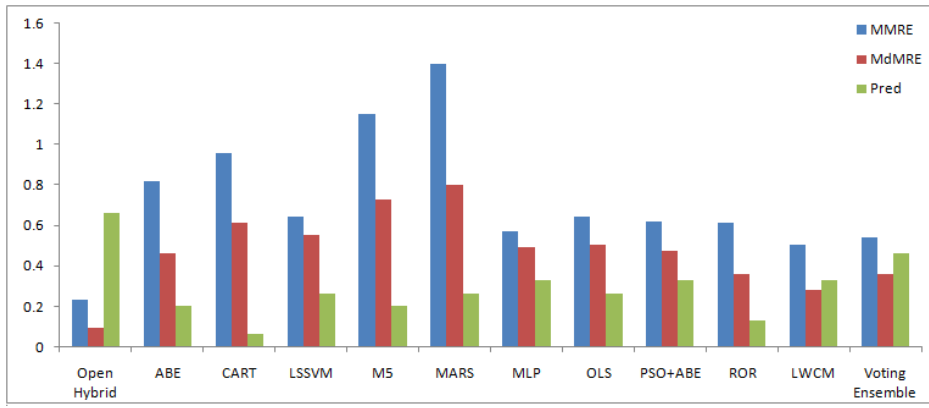


Figure 6. Comparing different methods in MMRE, MdMRE, and Pred on Kemerer

The proposed model was also tested on Desharnise and the results are presented in Table 5. 11 other methods were also tested on this dataset. The results of testing the proposed model were compared with those of other methods in terms of MMRE, MdMRE, and Pred. According to the results, the proposed method was nearly 35 %, 40 %, and 43 % more precise than the most precise method in MMRE (Voting Ensemble), the most precise method in MdMRE (Voting Ensemble), and the most precise method in Pred (Voting Ensemble), respectively. For a better comparison, the results were also shown on a diagram in Figure 7, indicating the proper capability of the proposed model.

Method	MMRE	MdMRE	Pred
Open Hybrid	0.26	0.17	0.59
LWCM	0.49	0.29	0.25
Voting Ensemble	0.4	0.28	0.41
ABE	0.74	0.4	0.28
CART	0.68	0.35	0.27
LSSVM	0.58	0.41	0.24
M5'	0.72	0.39	0.29
MARS	1.19	0.57	0.23
MLP	0.91	0.54	0.24
OLS	0.71	0.53	0.27
PSO + ABE	0.87	0.4	0.4
ROR	0.6	0.49	0.36

Table 5. The results of testing Desharnise in comparison with other methods

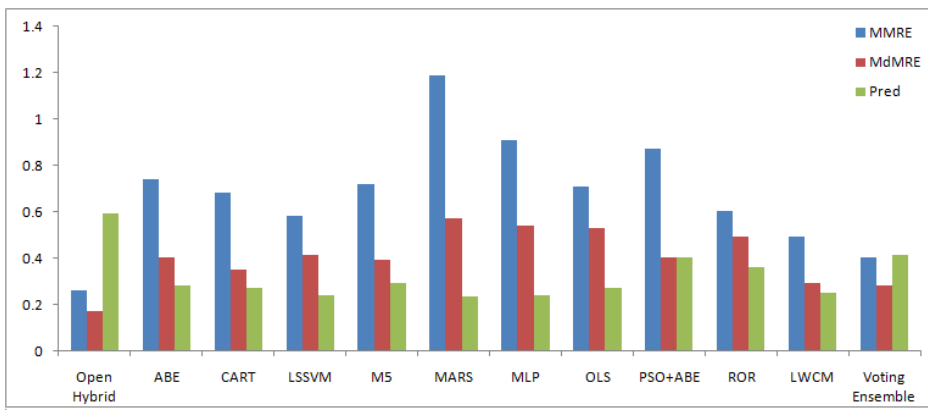


Figure 7. Comparing different methods in MMRE, MdMRE, and Pred on Desharnise

10 TEST RESULT ANALYSIS

Wilcoxon statistical test with two input statistical samples was used to determine the effectiveness of the proposed model. The output, P-value, shows the difference between the two input samples. A small P-value indicates the difference of the two samples, whereas a large value shows the similarity of both samples. A P-value less than 0.05 means a large difference between the two samples. Table 6 lists the Wilcoxon test results on MRE. It shows different P-values obtained from conducting the Wilcoxon test on the MRE of each method in comparison with the proposed model on different datasets. The results indicate the higher effectiveness of the proposed model in increasing estimation precision.

The box plot was employed for detailed analysis of the results of the proposed model. The box plot shows the MRE range obtained from testing the proposed

Method	Albrecht	Kemerer	Desharnise
ABE	0.0023	0.0021	0.0001
CART	0.0006	0.0014	7.95E−06
LWCM	0.019	8.00E−02	2.7E−02
Voting Ensemble	0.011	1.80E−02	4.20E−02
LSSVM	5.50E−04	4.70E−03	1.06E−05
M5'	0.0003	0.0012	1.17E−05
MARS	0.044	0.0018	5.36E−10
MLP	0.0005	3.40E−02	2.41E−08
OLS	0.013	0.027	1.58E−08
PSO + ABE	9.32E−05	5.60E−02	0.0002
ROR	5.00E−04	2.40E−03	1.19E−05

Table 6. P-values of Wilcoxon test

model on a dataset. Using this box plot, the MRE range and precision of the proposed model can easily be compared to those of other models. Figures 8, 9 and 10 show the box plots of MRE obtained from testing different methods on Albrecht, Kemerer, and Desharnise, respectively. As can be seen, the proposed model shows the lowest median and quartile range. All the results confirm the capability of the proposed model.

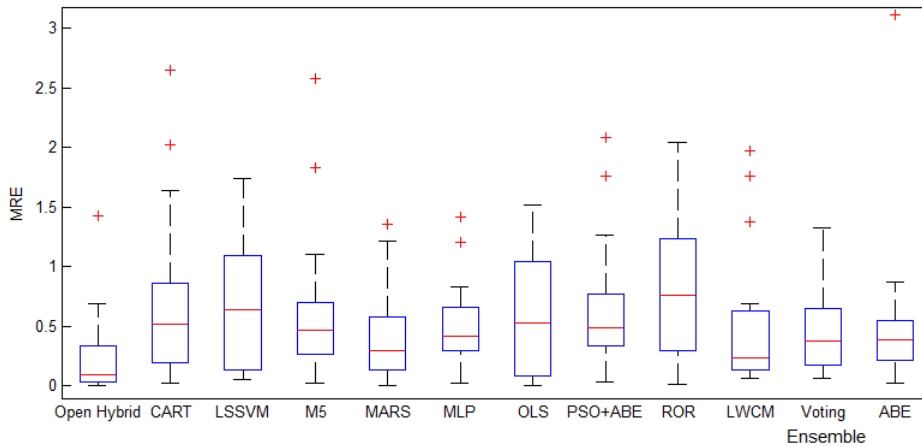


Figure 8. The box plot of MRE obtained by testing the proposed model on Albrecht

11 CONCLUSION

Software development cost estimation is an important issue resulting in an effective planning for software project management. Several methods and models have been proposed for this purpose. Despite acceptable precision of the proposed methods in

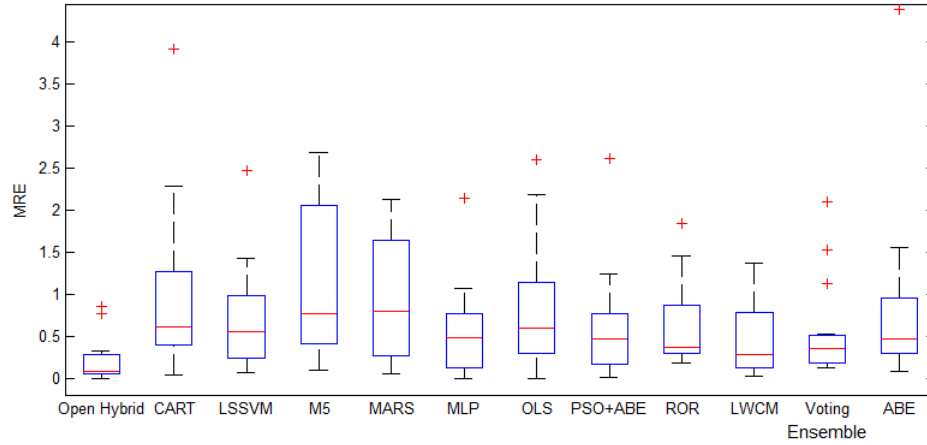


Figure 9. The box plot of MRE obtained by testing the proposed model on Kemerer

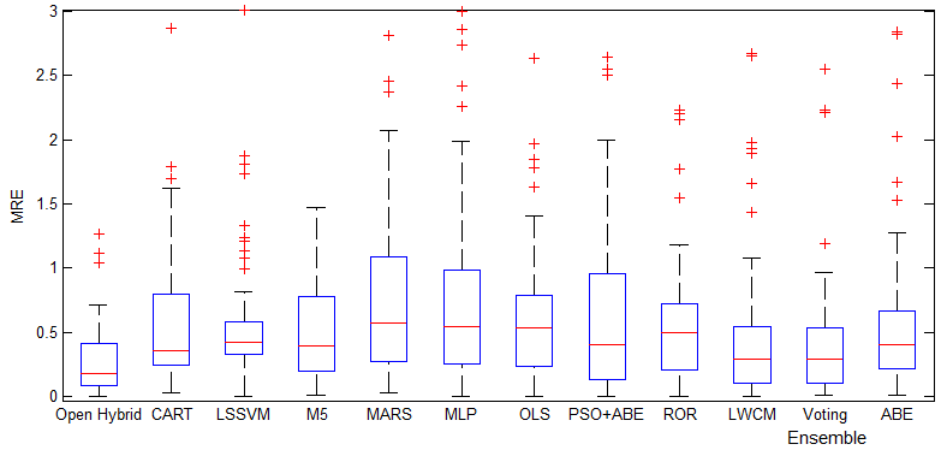


Figure 10. The box plot of MRE obtained by testing the proposed model on Desharnise

some cases, they fail operate accurately in other cases. The aim of this study was to combine these methods to take their advantages simultaneously to obtain a more accurate estimate. This idea led the authors to implement a model called the "open hybrid model".

The methods proposed in previous studies accurately estimate specific types of software projects when they depend on a specific estimation method. However, the challenge addressed here was to identify in what types of projects an estimation method can operate successfully, to what extent it can be successful, and how it is possible to take advantage of such a high level of precision. For this purpose, a new model called open hybrid model was developed in Section 5 based on the

FA. Different estimation methods can be added to the methods bank of this model. The proposed model is able to find the best rule for employing the methods in the methods bank using a variety of parameters and tools in the model.

The precision of the proposed model was evaluated by using three datasets of software projects. The results were compared with those obtained by testing 11 methods. The results were evaluated in terms of MMRE, MdMRE, and Pred. According to the results, the proposed model showed a high precision and ability to use capabilities of the methods existing in its methods bank. The results of tests were interpreted in detail in Section 9. Wilcoxon statistical test was conducted on the results to determine the effectiveness of the proposed model. The Wilcoxon results were analyzed in Section 10.

Regarding the other features of the proposed model, it was designed to include the new methods in its methods bank to use their advantages. Therefore, a new method with precise estimations in one or multiple datasets can be added to the methods bank. According to the results of multiple tests, the open hybrid model is able to intelligently identify a space in which a method can operate well. It can also effectively use every method under appropriate conditions. In future studies, more diverse and accurate methods and rules can be added to the methods and rules banks of this model.

REFERENCES

[1] Abnane, I.—Idri, A.—Abran, A.: Empirical Evaluation of Fuzzy Analogy for Software Development Effort Estimation. Proceedings of the Symposium on Applied Computing (SAC'17), ACM, 2017, pp. 1302–1304, doi: 10.1145/3019612.3019905.

[2] Andreou, A. S.—Papatheocharous, E.: Software Cost Estimation Using Fuzzy Decision Trees. 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, L'Aquila, Italy, 2008, pp. 371–374, doi: 10.1109/ase.2008.51.

[3] Arora, S.—Mishra, N.: Software Cost Estimation Using Artificial Neural Network. In: Pant, M., Ray, K., Sharma, T., Rawat, S., Bandyopadhyay, A. (Eds.): Soft Computing: Theories and Applications, Springer, Singapore, Advances in Intelligent Systems and Computing, Vol. 584, 2018, pp. 51–58, doi: 10.1007/978-981-10-5699-4_6.

[4] Azzeh, M.—Nassif, A. B.: A Hybrid Model for Estimating Software Project Effort from Use Case Points. Applied Soft Computing, Vol. 49, 2016, pp. 981–989, doi: 10.1016/j.asoc.2016.05.008.

[5] Bardsiri, A. K.—Hashemi, S. M.: Software Effort Estimation: A Survey of Well-Known Approaches. International Journal of Computer Science Engineering (IJCSE), Vol. 3, 2014, No. 1, pp. 46–50.

[6] Bardsiri, A. K.—Hashemi, S. M.: Empirical Evaluation of Different Machine Learning Methods for Software Services Development Effort Estimation Through Correlation Analysis. European Journal of Applied Sciences, Vol. 8, 2016, No. 4, pp. 257–269.

[7] Boehm, B. W.: Software Engineering Economics. New York, 1981.

[8] Boehm, B. W.—Abts, C.—Brown, A. W.—Chulani, S.—Clark, B. K.—Horowitz, E.—Madachy, R.—Reifer, D. J.—Steece, B.: Software Cost Estimation with Cocomo II. Prentice-Hall, 2000.

[9] Boehm, B. W.—Valerdi, R.: Achievements and Challenges in Cocomo-Based Software Resource Estimation. IEEE Software, Vol. 25, 2008, No. 5, pp. 74–83, doi: 10.1109/ms.2008.133.

[10] Cuadrado-Gallego, J. J.—Rodríguez-Soria, P.—Martín-Herrera, B.: Analogies and Differences Between Machine Learning and Expert Based Software Project Effort Estimation. 2010 11th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, IEEE, 2010, pp. 269–276, doi: 10.1109/snpd.2010.47.

[11] Sholiq—Dewi, R. S.—Subriadi, A. P.: A Comparative Study of Software Development Size Estimation Method: UCPabc vs Function Points. Procedia Computer Science, Vol. 124, 2017, pp. 470–477, doi: 10.1016/j.procs.2017.12.179.

[12] Dewi, R. S.—Subriadi, A. P.—Sholiq: A Modification Complexity Factor in Function Points Method for Software Cost Estimation Towards Public Service Application. Procedia Computer Science, Vol. 124, 2017, pp. 415–422, doi: 10.1016/j.procs.2017.12.172.

[13] Elish, M. O.: Assessment of Voting Ensemble for Estimating Software Development Effort. 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 2013, pp. 316–321, doi: 10.1109/cidm.2013.6597253.

[14] Cuadrado-Gallego, J. J.—Sicilia, M.-Á.: An Algorithm for the Generation of Segmented Parametric Software Estimation Models and Its Empirical Evaluation. Computing and Informatics, Vol. 26, 2007, No. 1, pp. 1–15.

[15] González-Ladrón-de-Guevara, F.—Fernández-Diego, M.—Lokan, C.: The Usage of ISBSG Data Fields in Software Effort Estimation: A Systematic Mapping Study. Journal of Systems and Software, Vol. 113, 2016, pp. 188–215, doi: 10.1016/j.jss.2015.11.040.

[16] Hsu, C.-J.—Rodas, N. U.—Huang, C.-Y.—Peng, K.-L.: A Study of Improving the Accuracy of Software Effort Estimation Using Linearly Weighted Combinations. 2010 IEEE 34th Annual Computer Software and Applications Conference Workshops, 2010, pp. 98–103, doi: 10.1109/compsacw.2010.27.

[17] Huang, J.—Li, Y.-F.—Xie, M.: An Empirical Analysis of Data Preprocessing for Machine Learning-Based Software Cost Estimation. Information and Software Technology, Vol. 67, 2015, pp. 108–127, doi: 10.1016/j.infsof.2015.07.004.

[18] Idri, A.—Azzahra Amazal, F.—Abran, A.: Analogy-Based Software Development Effort Estimation: A Systematic Mapping and Review. Information and Software Technology, Vol. 58, 2015, pp. 206–230, doi: 10.1016/j.infsof.2014.07.013.

[19] Khatibi, V.—Jawawi, D. N. A.: Software Cost Estimation Methods: A Review. Journal of Emerging Trends in Computing and Information Sciences, Vol. 2, 2011, No. 1, pp. 21–29.

[20] Khuat, T. T.—Le, M. H.: Applying Teaching-Learning to Artificial Bee Colony for Parameter Optimization of Software Effort Estimation Model. Journal of Engineering Science and Technology, Vol. 12, 2017, No. 5, pp. 1178–1190.

[21] Kocaguneli, E.—Menzies, T.: Software Effort Models Should Be Assessed via Leave-One-Out Validation. Journal of Systems and Software, Vol. 86, 2013, No. 7, pp. 1879–1890, doi: 10.1016/j.jss.2013.02.053.

[22] Kocaguneli, E.—Menzies, T.—Keung, J. W.: On the Value of Ensemble Effort Estimation. IEEE Transactions on Software Engineering, Vol. 38, 2012, No. 6, pp. 1403–1416, doi: 10.1109/TSE.2011.111.

[23] Kultur, Y.—Turhan, B.—Bener, A. B.: ENNA: Software Effort Estimation Using Ensemble of Neural Networks with Associative Memory. Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '08/FSE-16), 2008, pp. 330–338, doi: 10.1145/1453101.1453148.

[24] Kumari, S.—Pushkar, S.: Performance Analysis of the Software Cost Estimation Methods: A Review. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, 2013, No. 7, pp. 229–238.

[25] Malgonde, O.—Chari, K.: An Ensemble-Based Model for Predicting Agile Software Development Effort. Empirical Software Engineering, Vol. 24, 2019, No. 2, pp. 1017–1055, doi: 10.1007/s10664-018-9647-0.

[26] Mensah, S.—Keung, J.—Bosu, M. F.—Bennin, K. E.: Duplex Output Software Effort Estimation Model with Self-Guided Interpretation. Information and Software Technology, Vol. 94, 2018, pp. 1–13, doi: 10.1016/j.infsof.2017.09.010.

[27] Moosavi, S. H. S.—Bardsiri, V. K.: Satin Bowerbird Optimizer: A New Optimization Algorithm to Optimize ANFIS for Software Development Effort Estimation. Engineering Applications of Artificial Intelligence, Vol. 60, 2017, pp. 1–15, doi: 10.1016/j.engappai.2017.01.006.

[28] Ochieng, N.—Mwangi, W.—Ateya, I.: Optimizing Computer Worm Detection Using Ensembles. Security and Communication Networks, Vol. 2019, 2019, Art. No. 4656480, doi: 10.1155/2019/4656480.

[29] Pahariya, J. S.—Ravi, V.—Carr, M.: Software Cost Estimation Using Computational Intelligence Techniques. 2009 World Congress on Nature and Biologically Inspired Computing (NaBIC), IEEE, 2009, pp. 849–854, doi: 10.1109/nabic.2009.5393534.

[30] Pandey, P.: Analysis of the Techniques for Software Cost Estimation. 2013 Third International Conference on Advanced Computing and Communication Technologies (ACCT), IEEE, 2013, pp. 16–19, doi: 10.1109/acct.2013.13.

[31] Pospieszny, P.—Czarnacka-Chrobot, B.—Kobylinski, A.: An Effective Approach for Software Project Effort and Duration Estimation with Machine Learning Algorithms. Journal of Systems and Software, Vol. 137, 2018, pp. 184–196, doi: 10.1016/j.jss.2017.11.066.

[32] Puspaningrum, A.—Sarno, R.: A Hybrid Cuckoo Optimization and Harmony Search Algorithm for Software Cost Estimation. Procedia Computer Science, Vol. 124, 2017, pp. 461–469, doi: 10.1016/j.procs.2017.12.178.

[33] Prasad Reddy, P. V. G. D.—Hari, C. V. M. K.—Srinavasa Rao, T.: Multi Objective Particle Swarm Optimization for Software Cost Estimation. International Journal of Computer Applications, Vol. 32, 2011, No. 3, pp. 13–17.

[34] Rastogi, H.—Dhankhar, S.—Kakkar, M.: A Survey on Software Effort Estimation Techniques. 2014 5th International Conference – Confluence the Next Generation Information Technology Summit (Confluence), IEEE, 2014, pp. 826–830, doi: 10.1109/confluence.2014.6949367.

[35] Salehi, M.—Razmara, J.—Lotfi, S.: A Novel Data Mining on Breast Cancer Survivability Using MLP Ensemble Learners. The Computer Journal, Vol. 63, 2020, pp. 435–447, doi: 10.1093/comjnl/bxz051.

[36] Shekhar, S.—Kumar, U.: Review of Various Software Cost Estimation Techniques. International Journal of Computer Applications, Vol. 141, 2016, No. 11, pp. 31–34, doi: 10.5120/ijca2016909867.

[37] Sigweni, B.—Shepperd, M.: Feature Weighting Techniques for CBR in Software Effort Estimation Studies: A Review and Empirical Evaluation. Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE '14), ACM, 2014, pp. 32–41, doi: 10.1145/2639490.2639508.

[38] Silva, J.—Borré, J. R.—Piñeres Castillo, A. P.—Castro, L.—Varela, N.: Integration of Data Mining Classification Techniques and Ensemble Learning for Predicting the Export Potential of a Company. Procedia Computer Science, Vol. 151, 2019, pp. 1194–1200, doi: 10.1016/j.procs.2019.04.171.

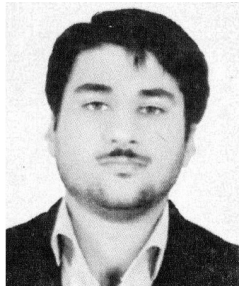
[39] Song, L.—Minku, L. L.—Yao, X.: The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. Proceedings of the 9th International Conference on Predictive Models in Software Engineering (PROMISE '13), ACM, 2013, Art. No. 9, doi: 10.1145/2499393.2499394.

[40] Wani, Z. H.—Quadri, S. M. K.: Software Cost Estimation Based on the Hybrid Model of Input Selection Procedure and Artificial Neural Network. Artificial Intelligent Systems and Machine Learning, Vol. 10, 2018, No. 1, pp. 18–24.

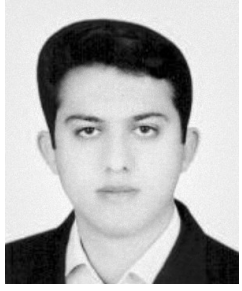
[41] Wen, J.—Li, S.—Lin, Z.—Hu, Y.—Huang, C.: Systematic Literature Review of Machine Learning Based Software Development Effort Estimation Models. Information and Software Technology, Vol. 54, 2012, No. 1, pp. 41–59, doi: 10.1016/j.infsof.2011.09.002.

[42] Wu, D.—Li, J.—Liang, Y.: Linear Combination of Multiple Case-Based Reasoning with Optimized Weight for Software Effort Estimation. The Journal of Supercomputing, Vol. 64, 2013, No. 3, pp. 898–918, doi: 10.1007/s11227-010-0525-9.

[43] Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (Eds.): Stochastic Algorithms: Foundations and Applications (SAGA 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5792, 2009, pp. 169–178, doi: 10.1007/978-3-642-04944-6_14.



Amin Moradbeiky received his B.Sc. from University of Sistan and Baluchestan, Iran in 2009 and M.Sc. from Islamic Azad University, Kerman Branch, Iran in 2014, both in engineering of information technology. He is currently Ph.D. student of software engineering in Islamic Azad University, Kerman Branch, Iran. His research interests are soft computing techniques, software test and software measurement.



Vahid Khatibi Bardsiri is a lecturer at the Department of Computer Science, Islamic Azad University, Bardsir Branch, Iran. He received his B.Sc. and M.Sc. degrees in software engineering from Ferdowsi University of Mashhad, Iran in 2002 and from Science and Research Branch of Islamic Azad University, Iran in 2004, respectively. He received his Ph.D. in the area of software development effort estimation at Universiti Teknologi Malaysia (UTM) in 2013. He is a senior member of International Association of Computer Science and Information Technology (IACSIT). His research interests are agile software development methods, soft computing techniques and software measurement.

Mehdi Jafari received his B.Sc. degree in electronic engineering and M.Sc. degree in communication systems from Shiraz University of Technology, Iran, in 1992 and 1996, respectively. During 1992–1998 he stayed in the Signal Processing Research Laboratory, Ministry of Telecommunications of Iran. He received Ph.D. degree in communication systems in Science and Research Branch of Islamic Azad University, Iran in 2008. He is currently Head of Electrical Engineering Department of Islamic Azad University, Kerman Branch, Iran. His main research interests are artificial intelligence, image and video processing, pattern recognition and prediction.