**SOFTWARE EFFORT ESTIMATION USING MODIFIED FUZZY C MEANS CLUSTERING AND HYBRID ABC-MCS OPTIMIZATION IN NEURAL NETWORK**

**Abstract:**

In a software development process, effective cost estimation is the most challenging activity. Software effort estimation is a crucial part of cost estimation. Management cautiously considers effort and benefits of software before committing the required resources to that project or order for a contract. Unfortunately it is difficult to measure such preliminary estimation as it has only little information about the project at an early stage. In this paper, a new approach has been proposed which is based on reasoning by soft computing approach in order to calculate the effort estimation of the software. In this approach, at first the datasets are clustered in order to create the rules. In our proposed approach we have utilized Modified Fuzzy C Means Clustering (MFCM) for clustering the dataset. Once the clustering is done, various rules are obtained and these rules are given as the input to the neural network. Here we have modified the neural network by incorporating optimization algorithms. The optimization algorithms employed here are Artificial Bee Colony (ABC) algorithm, Modified Cuckoo Search (MCS) algorithm and Hybrid ABC-MCS algorithm. Hence we obtain three optimized set of rules which are used for the effort estimation process. The performance of the proposed model is investigated using parameters such as Mean Absolute Relative Error (MARE) and Mean Magnitude of Relative Error (MMRE).

**1. Introduction:**

With the popularization of information engineering, customers realize the importance of project management and control in software project gradually. This requires accurate enough effort estimate and quality guarantee which provided by the software developers [7]. Accurately predicting software development effort is a critical concern of many organisations even today. Underestimating development cost and schedule can have a detrimental impact on both the functionality and quality of software products and therefore on the developer’s reputation and competitiveness. Software project cost and effort estimation is an increasingly important field due to the overwhelming role of software in today’s global market. However, there is no optimal approach to accurately predict the effort needed for developing a software system. Usually, the information gathered at the early stages of software system development is insufficient for providing a precise effort prediction [1]. Effort estimation methods could be based on subjective expert judgement or formal estimation models. Formal models use functional dependency to estimate effort using some value that quantifies project size [8].

To manage a software project effectively, accurate and reliable software effort estimation is critical activity. During the last decade, a number of software effort estimation methods with different concepts and approaches to combining the existing effort estimation methods have been developed and evaluated using historical project data [2]. A good estimation of size and effort available right from the start in a project gives the project manager confidence about any future course of action, since many of the decisions made during development depend on, or are influenced by, the initial estimations. Hence, the effort estimation is one of the most crucial steps of planning and management of a software project [3]. Development effort is considered to be one of the major components of software costs, particularly as regards global development, and it is usually the most challenging effort to predict. Several methods with which to support project managers when estimating software development effort for GSD projects have been proposed in the last few years [9].

In order to provide precise estimates and decrease the estimation error, several cost estimation models have been proposed and developed. Among these models, Constructive Cost Model (COCOMO) is the most commonly used model in the software companies because of its capabilities and features for estimating the software effort in person-month (PM) at different development stages [4]. Basic model is usually applied in the early phase of software development. When more detailed project information can be obtained in the later phase, the intermediate and advanced models can be used for project estimation and measurement [5]. As far as effort estimation is concerned, a number of unsolved problems and errors still exist. Hence to obtain good results, it is essential to consider data from previous projects. The need of accurate estimation has been always important while determining the feasibility of a project in terms of cost benefit analysis [6].

The rest of the paper is organized as follows. Section II explains the researches related to our proposed method. Section III shows our proposed method being used for software effort estimation process. Section IV explains the result of the proposed methodology and finally Section V concludes our proposed method with suggestions given for future works.

**2. Related Work:**

A predictive model was required to be accurate and comprehensible in order to inspire confidence in a business setting. Both aspects have been assessed in a software effort estimation setting by previous studies. However, no univocal conclusion as to which technique was the most suited have been reached. Dejaeger et al. [11] have studied the issue by reporting on the results of a large scale benchmarking study. Different types of techniques are under consideration, including techniques inducing tree/rule-based models like M5 and CART, linear models such as various types of linear regression, nonlinear models (MARS, multilayered perceptron neural networks, radial basis function networks, and least squares support vector machines), and estimation techniques that do not explicitly induce a model (e.g., a case-based reasoning approach). Furthermore, the aspect of feature subset selection by using a generic backward input selection wrapper was investigated.

Producing accurate and reliable software effort estimation has always been a challenge for both academic research and software industries. Regarding this issue, data quality was an important factor that impacts the estimation accuracy of effort estimation methods. To assess the impact of data quality, Seo and Bae [12] have investigated the effect of eliminating outliers on the estimation accuracy of commonly used software effort estimation methods. Based on three research questions,they associatively analyzed the influence of outlier elimination on the accuracy of software effort estimation by applying five methods of outlier elimination (Least trimmed squares, Cook's distance, K-means clustering, Box plot, and Mantel leverage metric) and two methods of effort estimation (Least squares regression and Estimation by analogy with the variation of the parameters). Empirical experiments were performed using industrial data sets . In addition, the effect of the outlier elimination methods was evaluated by the statistical tests.

Software development effort estimation (SDEE) was the process of predicting the effort required to develop a software system. In order to improve estimation accuracy, many researchers have proposed machine learning (ML) based SDEE models (ML models) since 1990s. However, there have been no attempt to analyze the empirical evidence on ML models in a systematic way. Wen et al. [13] have aimed to systematically analyze ML models from four aspects: type of ML technique, estimation accuracy, model comparison, and estimation context. They performed a systematic literature review of empirical studies on ML model published in the last two decades.

Due to the increasing complexity of software development activities, the need for effective effort estimation techniques has arisen. Underestimation leads to disruption in the project’s estimated cost and delivery. On the other hand, overestimation causes outbidding and financial losses in business. Effective software effort estimation techniques enable project managers to schedule the software life cycle activities appropriately. Correctly assessing the effort needed to develop a software product was a major concern in software industries. Random forest (RF) technique was a popularly used machine learning technique that helps in improving the prediction values. Satapathy *et al.* [14] have precisely assessed the software projects development effort by utilising the use case point approach. The effort parameters are optimised utilising the RF technique to acquire higher prediction accuracy. Moreover, the results acquired applying the RF technique was compared with the multi-layer perceptron, radial basis function network, stochastic gradient boosting and log-linear regression techniques to highlight the performance attained by each technique.

To characterize the essential content of SEE data; i.e. the least number of features and instances required to capture the information within SEE data. If the essential content was very small then (1) the contained information must be very brief and (2) the value-added of complex learning schemes must be minimal. Kocaguneli et al.[15] have computed the Euclidean distance between rows (instances) and columns (features) of SEE data; then prunes synonyms (similar features) and outliers (distant instances); then assesses the reduced data by comparing predictions from (1) a simple learner using the reduced data and (2) a state-of-the-art learner (CART) using all data.

**3. Proposed methodology for software effort estimation:**

**3.1 Steps involved in our proposed scheme of effort estimation:**

In this paper, a new approach has been proposed which is based on reasoning by soft computing approach in order to calculate the effort estimation of the software. In this approach, at first the datasets are clustered in order to create the rules. In our proposed approach we have utilized Modified Fuzzy C Means Clustering (MFCM) for clustering the dataset. Once the clustering is done, various rules are obtained and these rules are given as the input to the neural network. Here we have modified the neural network by incorporating optimization algorithms. The optimization algorithms employed here are Artificial Bee Colony (ABC) algorithm, Modified Cuckoo Search (MCS) algorithm and Hybrid ABC-MCS algorithm. Hence we obtain three optimized set of rules which are used for the effort estimation process. The overall flow diagram of our proposed method is shown in fig 1below,

Error estimates (MMRE, MARE, MRE)

Input dataset

Generation of testcases

Clustering of testcases using MFCM

Classification Using Hybrid Neural Network

Optimization using ABC-MCS

Output effort value

Proposed Classifier

**Fig 1:** Proposed software effort estimation model.

**3.2. Test case generation:**

Test cases are employed to test all feasible combinations in the application and as well it offers the user to simply replicate the steps that were assumed to expose a defect that as identified during test. Test cases can be charted directly and obtained from use cases. Test cases can as well be obtained from system requirements. Moreover, when the test cases are produced early, Software Engineers can frequently discover ambiguities and inconsistencies in the requirements specification and design documents. This will absolutely get down the cost of building the software systems as faults are eradicated early during the life cycle. Test case generation is a method where the test cases are produced not based on an algorithm but based on the ones statement of the application. Classes will be checked and different test inputs will be offered to make sure for the faults in the application. The generated test cases will be fed to the advanced neural network for classification based on which the software quality will be predicted.

**3.3 Clustering of testcases using MFCM algorithm:**

The modified fuzzy c means algorithm is commonly used for clustering where the performance of the MFCM depends on the selection of initial cluster center or membership value. The MFCM algorithm starts with a set of initial cluster centers (or) arbitrary member ship values.

The MFCM algorithm assigns pixels to each category by using fuzzy memberships.

 (1)

where,

 - extracted testcase,

- cluster centre.

- constant value.

- weighted mean distance in cluster i and is given by,

 (2)

The membership function represents the probability that a pixel belongs to a speciﬁc cluster. In the FCM algorithm, the probability is dependent on the distance between the pixel and each individual cluster center in the feature domain. The membership functions and cluster centers are updated by the equations (3) and (4).

 (3)

The clusters centroid values is computed by using the equation (4)

 (4)

Repeat the algorithm until the coefficients' change between two iterations, for the given sensitivity threshold.

 (5)

In equation (5),  is a termination criterion between 0 and 1. Repeat the process until efficient clustering is reached. Thus from the MFCM the testcases are clustered.

**3.4 Rules selection using HNN with ABC-MCS:**

The Hybrid Neural Network is utilized to ascertain the testcase selection and it is trained by employing the various testcases employed in the proposed scheme of effort estimation. The hybrid artificial neural network is well trained by means of the extorted testcases. The innovative HNN is home to three input units, n hidden units and one output unit. The input of the neural network is the testcases we have generated. The neural network works making use of two phases, one is the training phase and the other is the testing phase.

***A. TRAINING PHASE***

In the training phase, the testcases are given as the input to the neural network. Initially, the nodes are given random weights. As the output is already known in the training phase, the output obtained from the neural network is compared to the original and weights are varied so as to reduce the error. This process is carried for every testcases so as to yield a stable system having weights assigned in the nodes.

***B. TESTING PHASE***

In the testing phase, the testcases are fed to the trained neural network having particular weights in the nodes and the output is calculated based on the trained dataset. In ordinary neural network the process will be stopped after testing. In the proposed hybrid neural network, for testing process we have incorporated the optimization algorithm inorder to optimize the weight used for testing. In our proposed method the weights are optimized with the help of the ABC-MCS Algorithm. By incorporating optimization process the selection accuracy will be improved there by providing better effort estimation process. The structure of the artificial neural network is illustrated in Fig.2.



w22

2

1

NH

C2







w22N

w222

w221

w21N

w211

N



w|M|2

w|M|1

w12

1

2

I2

w11



I1

1

Input layer

Hidden layer

Output layer

|M|

2

w212

C1

CN

I|M|

w21

**Fig 2:** Structure of Artificial Neural Network

**3.4.1 Proposed Artificial Bee Colony – Modified Cuckoo Search 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 [19]. 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 [18].

*3.4.1.1 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, number of food sources around the hive and the number of employed bees are 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  with  solutions, where each solution is the food source position and is the population size is generated. The solution representation can be given as  is an N-dimensional vector and  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.

*3.4.1.2Onlooker 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 () by an onlooker bee can be given as follows.

(6)



where,

- ﬁtness of the solution and

- 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 and a new position , the relationship can be given as



(7)  
Where,





- arbitrary number in the range[ −1, 1].

The position update equation interprets that a decrease in the deviation between the parameters of andleads to a decrease in the perturbation on the position  . 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.



(8)  
A time domain representation can be given for the position update equation by considering  as  when is taken as. Hence, we obtain

(9)



As  refers to discrete version of the derivative of order, we can write

(10)



.

The output from the onlooker bee phase is the applied to the modified cuckoo search algorithm for further optimization which can aid in better weight optimization. The Cuckoo search algorithm represents a meta-heuristic algorithm which owes its origin to the breeding conduct of the cuckoos and it is easy of implementation. There is a multitude of nests in the cuckoo search. Each egg signifies a solution and an egg of cuckoo corresponds to a novel solution. The novel and superior solution replaces the most horrible solution in the nest. Similar to modified neural network, we have also modified the ordinary cuckoo search algorithm by including the Gauss distribution in the Updation phase where levy flight equation is used. The gauss distribution adds better results of optimization when compared to the normal process. The modus operandi of the clustering procedure is shown as follows:

**Step 1: Initialization Phase**

The population (mi, where i=1, 2, n) of host nest is initiated arbitrarily.

**Step 2: Generating New Cuckoo Phase**

With the help of the levy flights a cuckoo is selected randomly which generates novel solutions. Subsequently, the engendered cuckoo is evaluated by employing the objective function for ascertaining the excellence of the solutions.

**Step 3: Fitness Evaluation Phase**

The fitness function is evaluated in accordance with Equations 11 and 12 shown hereunder, followed by the selection of the best one.





Where,

- signifies the selected population

- represents the total population

**Step 4: Updation Phase**

At the outset, the solution is optimized by the levy flights by employing the cosine transform. The quality of the novel solution is evaluated and a nest is selected arbitrarily from among them. If the quality of novel solution in the selected nest is superior to the previous solution, it is replaced by the novel solution (Cuckoo). Otherwise, the previous solution is treated as the best solution. The levy flights employed for the general cuckoo search algorithm is expressed by the Equation 13 shown below:



By suitably adapting Equation 13, levy flight equation using the gauss distribution is exhibited in Equation 14 here under:



Where,



 - represents the constants

K – Symbolizes the current generation

**Step 5: Reject Worst Nest Phase**

In this section, the worst nests are ignored, in accordance with their possibility values and novel ones are constructed. Subsequently, depending upon their fitness function the best solutions are ranked. Thereafter, the best solutions are detected and marked as optimal solutions.

**Step 6: Stopping Criterion Phase**

Till the achievement of the maximum iteration, the procedure is continued.

By deftly employing the above-mentioned optimization process we have obtained improved rate of effective testcase selection that aid in improved effort estimation of the software.

**4. Results and discussion:**

In our proposed method we have employed modified ABC MCS algorithm for the rules optimization inorder to find out the software effort estimation. 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 and the results for the proposed method along with that of the existing methods are compared inorder to show the effectiveness of the proposed method. The effort that are obtained using our proposed method is tabulated in the below table along with the actual effort. The MRE value for proposed method is then calculated based on the effort estimated and the actual effort being calculated.

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



where,

- Estimated effort

-Actual effort

**4.1 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 (18) given below in the performance evaluation section, MMRE for the efforts are determined for the execution time. The table 1 below shows the estimated and actual effort based on which the MRE value is estimated.

Table 1: Estimated and Actual effort with MRE value

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Actual effort** | **Estimated effort** | **MRE** |
| 1 | 7538.342 | 7392.341 | 0.019368 |
| 2 | 5037.401 | 4989.401 | 0.009529 |
| 3 | 797.177 | 953.176 | 0.19569 |
| 4 | 4926.516 | 4584.516 | 0.06942 |
| 5 | 3341.558 | 3498.557 | 0.04698 |
| 6 | 4195.771 | 4045.77 | 0.035751 |
| 7 | 3157.767 | 2908.767 | 0.078853 |
| 8 | 5510.152 | 5367.151 | 0.025952 |
| 9 | 4817.673 | 4967.673 | 0.03114 |
| 10 | 2445.469 | 2597.469 | 0.06216 |
| 11 | 5735.558 | 5892.558 | 0.02737 |
| 12 | 7249.34 | 7405.339 | 0.02152 |
| 13 | 1452.7 | 1604.7 | 0.10463 |
| 14 | 5002.478 | 5161.477 | 0.03178 |
| 15 | 5674.268 | 5328.268 | 0.060977 |
| 16 | 3098.382 | 2850.382 | 0.080042 |
| 17 | 4038.353 | 4188.352 | 0.03714 |
| 18 | 7007.879 | 7161.879 | 0.02198 |
| 19 | 7885.784 | 8036.784 | 0.01915 |
| 20 | 1633.202 | 1586.202 | 0.028778 |
| 21 | 9488.004 | 9640.004 | 0.01602 |
| 22 | 8337.024 | 8493.023 | 0.01871 |
| 23 | 7422.679 | 7381.679 | 0.005524 |
| 24 | 7847.99 | 8005.99 | 0.02013 |
| 25 | 7045.08 | 7196.08 | 0.02143 |
| 26 | 6071.397 | 6222.397 | 0.02487 |
| 27 | 5218.878 | 5076.878 | 0.027209 |
| 28 | 8758.911 | 8608.911 | 0.017125 |
| 29 | 6666.486 | 6319.486 | 0.052051 |
| 30 | 7795.007 | 7647.007 | 0.018987 |

**4.2 Performance Analysis:**

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



where,

- Estimated effort

-Actual effort.

The MMRE calculation for the estimated effort can be done using equation (18). The proposed method has accomplished better MMRE than other fuzzy – based works.



The graphical illustration of the obtained effort value from the proposed method is affixed in Fig 3 in which the actual effort is relatively higher than the estimated effort values.

**Fig 3:** Graphical representation of 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 values are measured in percentage. The obtained values are given in table 2 below,

|  |  |  |
| --- | --- | --- |
| **Parameters** | **Proposed Results (%)** | |
| **Before Optimization** | **After Optimization** |
| MMRE | 13.608 | 0.04101 |
| MARE | 65.639 | 1.23027 |

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

The performance improvement accomplished from post – optimization process is better illustrated using the following fig 4, where the error values obtained from prior optimization and post optimization are plotted.

**Fig 4:** Graphical representation of MMRE and MARE values before and after optimization.

The effectiveness of the proposed method is then proved by compariing it with existing methods through MMRE measurements obtained from proposed and existing methods. The values are tabulated in Table 3, where MMRE is given in percentage.

|  |  |
| --- | --- |
| **Methods** | **MMRE (%)** |
| Proposed Method | 0.04101 |
| ABC | 0.6428 |
| MCS | 0.1218 |
| Fuzzy method | 0.32651 |
| GA based method | 0.0947 |

**Table 3**: MMRE measurementsfor the proposed and existing methods.

Fig 5 portrays the graphical illustration of comparative results between the proposed method and the existing method based on MRE and MMRE measurements. Here, the comparison in terms of MMRE is made between the proposed system, our previous work where ABC is used and the existing methods given in [16] and [17]. The graphical illustration demonstrates that the proposed method outperforms the existing methods in terms of efficiency.

**Fig 5:** Graphical representation of comparison of MMRE value for proposed and existing methods.

**5. CONCLUSION.**

In this proposed work we have utilized Modified Fuzzy C Means Clustering (MFCM) for clustering the dataset. Once the clustering is done, various rules are obtained and these rules are given as the input to the neural network. Here we have modified the neural network by incorporating optimization algorithms. The optimization algorithms employed here are Artificial Bee Colony (ABC) algorithm, Modified Cuckoo Search (MCS) algorithm and Hybrid ABC-MCS algorithm. Hence we obtain three optimized set of rules which are used for the effort estimation process. The performance of the proposed model is investigated using parameters such as Mean Absolute Relative Error (MARE) and Mean Magnitude of Relative Error (MMRE).The experimental outcomes have demonstrated the proposed system outperform the existing method in estimating the software effort more precisely.

**Reference:**

1. Khaled Hamdan, Hazem El Khatib, John Moses and Peter Smith, "A Software Cost Ontology System for Assisting Estimation of Software Project Effort for Use with Case-Based Reasoning”, Innovations in Information Technology, 2006.
2. Yeong-Seok Seo, Kyung-A Yoon and Doo-Hwan Bae, "Improving the Accuracy of Software Effort Estimation based on Multiple Least Square Regression Models by Estimation Error-based Data Partitioning", In.proc.of 16th Asia-Pacific Software Engineering Conference,2009.
3. Rolan Abdukalykov, Ishrar Hussain, Mohamad Kassab and Olga Ormandjieva, "Quantifying the Impact of Different Non-Functional Requirements and Problem Domains on Software Effort Estimation",In.proc.of Ninth International Conference on Software Engineering Research, Management and Applications,2011.
4. Iman Attarzadeh, Amin Mehranzadeh and Ali Barati,"Proposing an Enhanced Artificial Neural Network Prediction Model to Improve the Accuracy in Software Effort Estimation", In.proc.of Fourth International Conference on Computational Intelligence, Communication Systems and Networks, 2012.
5. Wen-Tin Lee, Jonathan Lee,Kuo-Hsun Hsu and Jong Yih Kuo, "Applying Software Effort Estimation Model based on Work Breakdown Structure", In.proc.of Sixth International Conference on Genetic and Evolutionary Computing,2012.
6. Shashank Mouli Satapathy, Mukesh Kumar and Santanu Kumar Rath, "Class Point Approach for Software Effort Estimation using Soft Computing Techniques", ACM SIGSOFT Software Engineering ,Vol.39, No.3,pp.1-6,2014.
7. Zhang Dan, "Improving the Accuracy in Software Effort Estimation Using Artificial Neural Network Model Based on Particle Swarm Optimization", In.proc.of IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 2013.
8. Jovan Popovic, Dragan Bojic and Nenad Korolija,"Analysis of task effort estimation accuracy based on use case point size", IET Software, pp.1–8, 2015.
9. Manal El Bajta, "Analogy-based software development effort estimation in global software development", In.proc.of IEEE 10th International Conference on Global Software Engineering Workshops, 2015.
10. T.Wijayasiriwardhane, R.Lai and K.C.Kang, "Effort estimation of component-based software development – a survey", IET Software, Vol.5, No.2, pp.216–228, 2011.
11. Karel Dejaeger,David Martens and Bart Baesens,"Data Mining Techniques for Software Effort Estimation: A Comparative Study", IEEE Transactions on Software Engineering, Vol.38, No.2,2012.
12. Yeong-Seok Seo and Doo-Hwan Bae, "On the value of outlier elimination on software effort estimation research", Journal Empirical Software Engineering, Vol.18, No.4, 2013.
13. Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, Changqin Huang,"Systematic literature review of machine learning based software development effort estimation models", Journal Information and Software Technology , Vol.54,No.1,2012.
14. Shashank Mouli Satapathy, Barada Prasanna Acharya and Santanu Kumar 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.
15. 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.
16. 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.
17. Adriano L.I. Oliveira, Petronio L. Braga, Ricardo M.F. Lima and Marcio L. Cornelio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation”, Information and Software Technology, Vol.52, pp.1155–1166, 2010.
18. DervisKaraboga, BahriyeAkay, A comparative study of Artificial Bee Colony algorithm, Journal of Applied Mathematics and Computation, Vol. 214, Pp. 108–132, 2009.
19. DervisKaraboga and CelalOzturk, Fuzzy clustering with artificial bee colony algorithm, Journal of Scientific Research and Essays, Vol. 5, No. 14, pp. 1899-1902, 2010.