Received January 10, 2020, accepted February 8, 2020, date of publication March 13, 2020, date of current version April 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2980236

Ensembling Artificial Bee Colony With Analogy-Based Estimation to Improve Software Development Effort Prediction

MUHAMMAD ARIF SHAH[1,2], DAYANG NORHAYATI ABANG JAWAWI[1], MOHD ADHAM ISA[1], MUHAMMAD YOUNAS[3], ABDELZAHIR ABDELMABOUD[4], AND FAUZI SHOLICHIN[1]

[1]Department of Software Engineering, Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia
[2]Department of IT and Computer Science, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur 22620, Pakistan
[3]Department of Computer Science, Government College University Faisalabad, Faisalabad 38000, Pakistan
[4]Department of Information Systems, King Khalid University, Abha 62529, Saudi Arabia

Corresponding authors: Muhammad Arif Shah (arif.websol@gmail.com) and Muhammad Younas (younas.76@gmail.com)

This work was supported by the Universiti Teknologi Malaysia (UTM) through UTM-TDR Research under Grant 06G23.

ABSTRACT Analogy-Based Estimation (ABE) is one of the promising estimation models used for predicting the software development effort. Researchers proposed different variants of the ABE model, but still, the most suitable procedure could not be produced for accurate estimation. In this study, an artificial Bee colony guided Analogy-Based Estimation (BABE) model is proposed which ensembles Artificial Bee Colony (ABC) with ABE for accurate estimation. ABC produces different weights, out of which the most appropriate is infused in the similarity function of ABE during the stage of model training, which are later used in the testing stage for evaluation. There are six real datasets utilized for simulating the model procedure. Five of these datasets are taken from the PROMISE repository. The predictive performance is improved for BABE over the existing ones. The most significant of its performance is found on the International Software Benchmarking Standards Group (ISBSG) dataset.

INDEX TERMS Analogy based estimation, cost estimation, artificial bee colony, software development, project management.

I. INTRODUCTION

Estimating the software development effort is a paramount and chaotic activity in project management. Planning and controlling a software project become unmanageable without accurate estimates. The effort prediction models have been successful in estimating the development effort of a software project as compared to the unreliable estimates practiced in the industry. Time and budget are the nitty-gritty of a software project due to the rapid evolution of software-dependent hardware technology and rapid change in customer requirements. Moreover, as opposed to other types of projects, the nature of software projects is intangible due to which, the effort cannot be measured until work on the project is initiated [1].

Researchers have been working for decades to accurately estimate software development effort for effective planning and controlling software projects. The estimation process progressed from very simple assumptions to complex techniques. Today, the estimation methods can be divided into two broader categories of Algorithmic models and Non-Algorithmic models. According to Jones [2], the manual rule of thumb is the very first algorithmic estimation method which was initiated in 1950. The users of quality software increased due to which regression techniques and linear equations-based estimation models were brought forward [3]. According to Jones [2], Barry Boehm, Joe Aron, and Larry Putnam are the founding fathers of estimation methods for software. Interactive Productivity and Quality (IPQ) was the first automated estimation tool presented in 1973 by the researchers of IBM. Barry Boehm as one of the most contributor to estimation model produced COCOMO and many other estimation algorithms which are quite prominent until yet [4]. Estimation of Resources-Software Estimating Model (SEER-SEM) and Putnam Lifecycle Management (SLIM) models

The associate editor coordinating the review of this manuscript and approving it for publication was Xiao Liu.

also followed the guidelines of COCOMO [3]. Function Point (FP) was another method introduced to measure and predict the size of the software project [5]. Prior to FP, the Lines Of Code (LOC) estimation method was used in all the studies but FP was found a much more accurate and general-purpose estimation method because it focuses on measuring the functionalities rather than the number of lines or statements in a program. The swift advancements in software development methodologies guided to bring about COCOMO II which is an extended version of COCOMO [6]. The COCOMO II model used FP to measure the size of a software project based on functionalities rather LOC.

Non-Algorithmic estimation models use the completed projects for estimation as opposed to algorithmic models. Algorithmic models are usually unable to handle the intangible and dynamic nature of a software project. It becomes burdensome and difficult to estimate the development effort of software due to lack of information in the early stage of a project which led researchers to develop non-algorithmic estimation models. The expert judgment methods were proffered as some of the researchers felt the need for expert opinion in the estimation process [7]. There were several studies observed to have the guidelines of expert judgment analyzed [8], [9].

Classification And Regression Tree (CART) is another very prominent non-algorithmic estimation method that is used to construct a regression tree based on the past project information where the amount of effort applied is represented by leaves. The targeted project's features determine the path to be traversed from root to leaf.

Analogy-Based Estimation (ABE) is a commonly used non-algorithmic model that estimates the cost of a targeted project by comparing and finding the most related project from the pool of past projects [10]. Comparison among projects is performed based on similar features such as FP, the type of application, LOC. Some of the researchers showed interest in considering interval rather fixed value for estimation due to disparities in estimates.

Different soft-computing techniques such as Neural Network, Fuzzy Logic, Particle Swarm Optimization (PSO), Nearest Neighbors, Genetic Algorithm (GA), etc. are also adopted for increasing the estimation accuracy of software development effort [1], [11]–[17]. The optimization techniques focus to improve the feature selection or attribute weighting in the similarity function of ABE. Most of the optimization algorithms are inspired by nature, such as PSO mimics the bird flocking and fish schooling behavior. Firefly Algorithm is inspired by the attraction behavior of fireflies matting. Ant Colony Optimization is motivated by the behavior of simulated ants. Artificial Bee Colony (ABC) algorithm represents the bees' behavior of food search. According to Cao *et al.* [18] ABC is a popular optimization algorithm due to better performance of search optimization and fewer control parameters. According to a comparative study conducted by Bao and Zeng [19], ABC performs better than Differential Evaluation (DE), PSO, GA in terms of accuracy. This study focuses on optimizing the feature weights to improve ABE by ensembling ABC with it as to the best of our knowledge, the impact of ABC on feature weighting of ABE has not been studied.

The background and related works are presented in the introduction section, moreover, the related work of ABE and ABC is presented in their respective sections. The rest of the paper is organized as Section II and III, explain the ABE and Artificial Bee Colony (ABC) algorithm respectively. Section IV includes details of the proposed model. The results are shown in Section V followed by the conclusion in Section VI.

II. ESTIMATION BY ANALOGY (ABE)

ABE or EBA was introduced as the non-algorithmic estimation method by Shepperd and Schofield [10]. It estimates the effort of a new project by comparing it with the historical projects. There are usually four parts of ABE,

1) Historical Projects
2) Similarity Function
3) Solution Function
4) Associated Retrieval Rule

Each of which can be described as:

1) Collecting the data of previous projects to form a historical dataset.
2) Selecting the project's appropriate features.
3) Retrieving the data of past projects to find similarities with the target project. The weighted Manhattan Distance and Weighted Euclidean Distance are usually preferred at this stage.
4) To estimate the software development effort of the target project.

A. SIMILARITY FUNCTION

In ABE, similarity function is used to compare the features of two projects. Euclidean Similarity (ES) and Manhattan Similarity (MS) are the two prominent similarity functions used by ABE to find out the similarity between target and past projects [10]. The ES is shown in Equation 1.

$$Sim\left(p,p'\right)=\frac{1}{\left|\sqrt{\sum_{i=1}^{n} w_i Dis\left(fi,fi'\right)+\delta}\right|}\delta=0.0001$$

$$Dis\left(f_i,f_i\right)=\begin{Bmatrix} (f_i-f_i') & if\ f_i\ and\ f_i'\ are\ numerical\ or\ ordinal \\ 0 & if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i=f_i' \\ 1 & if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i\neq f_i' \end{Bmatrix} \quad (1)$$

where $\boldsymbol{p}$ and $\boldsymbol{p}'$ represent the projects to be compared, $\mathbf{w}_i$ is the weight allocated to the features which can range between $\mathbf{0}$ to $\mathbf{1}$. δ is used to retrieve a non-zero result. $\boldsymbol{f_i}$ and $\boldsymbol{f_i'}$ represent the project features while n determines the number of features.

There are many similarities between MS and ES, but MS calculates the absolute difference between features.

MS function is shown in Equation 2.

$$Sim(p, p') = \frac{1}{\left[\sum_{i=1}^{n} w_i Dis(f_i, f_i') + \delta\right]} \delta = 0.0001$$
$$Dis(f_i, f_i') = \begin{cases} |f_i - f_i'| & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases} \quad (2)$$

where p and p' represent the projects to be compared, $\mathbf{w_i}$ is the weight allocated to the features which can range between **0** to **1**. δ is used to retrieve a non-zero result. f_i and $f\,i'$ represent the project features while n determines the number of features.

B. SOLUTION FUNCTION

Once the K most similar projects are chosen, it becomes possible to predict or estimate the effort of target project according to the selected attributes or features. The Closest Analogy [20], the median [21], average and the inverse weighted mean of the most similar project are the most common solution functions [22]. Median refers to the median or effort for $K > 2$ similar projects, mean refers to the average of effort for $K > 1$. In estimation, the portion of each project is adjusted by the inverse distance weighted mean by Equation 3.

$$C_P = \sum_{K=1}^{K} \frac{Sim(p, p_k)}{\sum_{i=1}^{n} Sim(p, p_i)} C_{p_k} \quad (3)$$

where the new project is depicted by p, p_k shows the most similar project at kth, Cp_k illustrates the value of effort of k_{th} p_k and the total number of the projects is denoted by K.

III. ARTIFICIAL BEE COLONY

Dervis Karaboga developed the Artificial Bee Colony (ABC) algorithm which reflects the honey foraging behavior of honey bees [46]. In ABC, bees show up a swarm or collective intelligence to solve a particular optimization problem using employed bees, onlooker bees and scout bees [47], [48]. Employed bees are responsible for the food sources allocated where each bee is associated with an individual food source that makes the number of food sources and employed bees equal. Employed bee dances in the hive after it visits the food source. The food sources are exploited by the employed bees and the information of the nectar amount is passed on to the onlooker bees. Onlooker bees and employed bees remain same in number. The information received from employed bees is utilized to exploit the food sources and the neighborhood. This continues until the food sources are exhausted. Once the food source is exhausted, its employed bee becomes a scout to search for the newly available food sources. The quality of the solutions of the food source is determined by the nectar information. The onlooker bees mostly select the food source based on the increased amount of nectar information. The fundamental ABC steps can be seen in Figure 1.

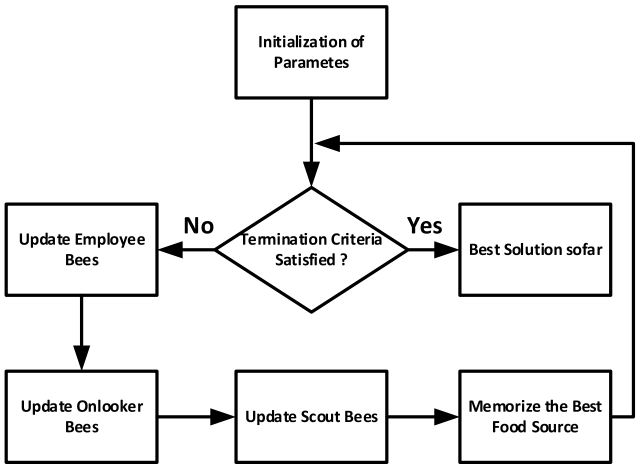

Initialization of Parametes
Termination Criteria Satisfied ?
No
Yes
Update Employee Bees
Best Solution sofar
Update Onlooker Bees
Update Scout Bees
Memorize the Best Food Source

FIGURE 1. The workflow of artificial bee colony algorithm.

IV. RELATED WORKS

Numerous studies focused on the correlation coefficient to improve ABE which is utilized for feature selection and weighting. Features of the projects with strong correlation are considered as the most similar features and are given high weight whereas low weights are assigned to the features with low correlations and are considered as less similar. The uncorrelated features are marked as dissimilar and are eliminated from the set of historical projects. This method has been shown as a positive improvement to ABE by some of the studies [23], [24]. Another weighting technique, known as Rough Set Analysis (RSA) [25] is also used for feature selection to improve the performance of ABE [26]–[28]. RSA analyzes dependencies between features, where the condition features are considered as independent and decision features depict the role of dependent features. In RSA, feature dependency analysis produces numerous subsets of features called reducts or classes. The most relevant feature is retrieved by taking the intersection of all the reducts. The weighting criteria of RSA is developed based on the existence of features in core sets, and the frequency of decision rules and reducts. Gray Theory is also one of the non-algorithmic estimation approaches, in which gray depicts the concept of fuzziness, known and unknown information is represented by white and black respectively [29]. It is a statistical method to find the level of similarity between two projects by comparing their features. Since it also practices a comparison method, it was adopted to improve the ABE performance [14], [30], [31].

Solution function is one of the vital parts of ABE because it significantly affects the performance of estimation accuracy. Accordingly, several researchers tried to apply adjustment expressions to enhance the solution function. Adjustment expressions are used to refine the solution function for better estimates [11], [32], [33].

Researchers greatly utilized Search-Based Software Engineering (SBSE) using metaheuristic searching methods such as Genetic Algorithm (GA), Simulated Annealing, Particle Swarm Optimization (PSO) [34] and Differential

Evaluation (DE) [16], [17] for optimizing the performance of ABE. Further studies that used SBSE for increasing the software development effort prediction. There are different optimization methods used to adjust feature weights in the solution function of ABE. The most common optimization method used for calculating feature weights in the ABE model is GA [11], [26]–[28], [35]. Huang and Chiu [35] used GA to find the best parameter in his defined linear and non-linear equations. The parameters involved in the linear and non-linear equations was concluded as a positive improvement in the ABE's performance. Kumari and Pushkar [36] used GA for multi-criteria-based project selection. There have been combined several techniques with GA to enhance the effort estimation accuracy such as Gray Relational Similarity technique [30], regression methods [37] and Linear adjustment [33].

PSO has also been used by many researchers to improve the software development effort estimation. Lin and Tzeng [38], Sheta *et al.* [39], and Hari and Reddy [40] used PSO to improve the performance of the COCOMO estimation model. PSO is considered computationally better than GA in selective cases as claimed by Bardsiri *et al.* [34]. Wu *et al.* [41] and Bardsiri *et al.* [42] employed PSO for weight optimization in the similarity measures of the ABE model. Bardsiri *et al.* [42] used PSO in combination with GA, Neural Networks, GA and Fuzzy Logic to design a localized effort estimation model Liu *et al.* [43] adopted PSO to improve estimation by minimizing the errors at the training stage. Azzeh *et al.* [44] used PSO for the identification of optimal decision variable so the tradeoff among different evaluation measures is presented. PSO has shown good results due to high convergence but the main goal in software development effort estimation is to improve the accuracy. Artificial Bee Colony is said to have good results in optimization accuracy then PSO [19].

The optimization targets or the fitness functions hold a significant role in estimation due to the highly uncertain nature of software projects. A detailed study on the impact of fitness function on the estimation accuracy was conducted by Ferrucci *et al.* [45] which concluded that the estimation accuracy could significantly be improved by selecting the appropriate and optimized performance metrics.

Artificial Bee Colony (ABC) has been used for a number of optimization problems in different domains. In some recent studies, ABC has been adopted and modified for software cost estimation. Sharma and Pant [49] Introduced the Halton sequence for initial distribution in ABC and applied the algorithm to handle the problems of predicting the cost model parameters. Gharehchopogh and Dizaji [50] proposed a hybrid of ABC, Chaos Optimization and Bees Colony for software cost estimation in contrast to the traditional COCOMO model. Khuat and Le [51] used Teaching-Learning with ABC for parameter optimization to overcome the limitations of the COCOMO II model. Gharehchopogh *et al.* [52] tried to measure the depended among the COCOMO factors using ABC for predicting the effort and cost of software projects. Pratama and Sarno [53] proposed to use ABC for optimizing and calibrate parameters of COCOMO II for accurate effort estimation. Rao *et al.* [54] hybridized ABC with local search to propose Multi-Layer Perceptron Neural Network (MLPNN) for classifying ranked attributes in the COCOMO dataset. Khuat and Le [55] proposed a directed artificial bee colony for tuning the model parameters values based on the actual effort of NASA datasets. Gharehchopogh *et al.* [56] hybridized ABC with Genetic Algorithm (GA) and claimed that introducing GA based ABC improves the performance of software estimation models.

Feature selection or optimizing the attribute selection from large datasets is one of the applications of ABC. Gharehchopogh *et al.* [56] combined ABC with Ant Colony Optimization (ACO) to reduce the global search and eliminate the stagnation behavior of ants by employed bees. Bees exploitation is used by ants to find the best subset of feature and the best ant; the subsets of features produced by ants are adopted by the bees as food sources. Reisi *et al.* [57] proposed feature weighted artificial bee colony for clustering the big data to improve the clustering quality by allocating different importance values to each feature. Hancer *et al.* [58] used fuzzy mutual information to design multi-objective ABC for selecting and filtering out the right attributes. Yavuz and Aydin [59] presented an angle modulation technique used with ABC for eliminating the problem of subset feature selection by reducing the high dimensional optimization to 4-dimensional continues optimization; the study claimed to have improved the classification accuracy. Kuo *et al.* [60] combined Support Vector Machine and Decision Tree with ABC for optimizing feature selection and parameters for rule extraction Ozturk *et al.* [61] implemented the new solution generation procedure through gene inspired components to enhance discreet ABC for handling the selection of similar cases. Wang *et al.* [62] introduced equivalence word set to ABC for feature selection and to filter redundant information from a large pool of data. Wang *et al.* [63] improved the initialization and scout bee steps of ABE for optimizing the classification performance and to optimize feature subset selection. Hancer *et al.* [64] proposed binary and continuous representations of ABC for feature selection using multi-objective artificial bee colony which was integrated with genetic operators and sorting procedures.

V. ARTIFICIAL BEE COLONY GUIDED ANALOGY-BASED ESTIMATION (BABE)

This section explains the working of our model. The training stage of BABE is discussed in Section *B* and the testing stage is explained in Section *C*. A group of training set projects and basic projects are infused in our model to train the model for predicting effort ofthe target project. The ABE part of it is adjusted by feature weights which are considered as the candidate parameters and the ABC part calculates the feature weight to reduce the value of MMRE which indicates lowering the estimation error for the training set. The trained BABE model received the test data for development effort estimation. Figure 2 and Figure 3 presents the flow of the training stage and testing stage respectively, whereas Algorithm 1, shows the Pseudo-code of BABE.

A. PERFORMANCE MEASURE

There have been used several methods for evaluating the performance of estimation methods in primary studies. According to the results of this study, the most prominent of them are Relative Error (RE), Mean Relative Error (MRE), Mean Magnitude of Relative Error (MMRE), and Percentage of Prediction (PRED). The computational equation of RE, MRE, MMRE, and PRED are shown in Equation 4, 5 and 6 simultaneously [10].

$$RE = (Estimated - Acutal)/Actual \tag{4}$$
$$MRE = |Estimated - Actual| /(Actual) \tag{5}$$
$$MMRE = \sum MRE/N \tag{6}$$
$$PRED\,(X) = \frac{A}{N} \tag{7}$$
$$AE = Estimated - Acutal \tag{8}$$
$$EF = \frac{PRED\,(25)}{1 + MMRE} \tag{9}$$
$$MAR = \frac{\sum_{i=1}^{N} AEi}{N} \tag{10}$$
$$SA = 1 - \frac{MAR}{\overline{MARp_0}} \tag{11}$$
$$\Delta = \frac{MAR - \overline{MARp_0}}{Sp_0} \tag{12}$$

In Equation 6 and 7, N represents the number of projects, A represents the projects with MRE >= X. The level of X is usually kept at 0.25 in software development effort estimation. The main aim of all the effort estimation models is to increase PRED and decrease MMRE. Evaluation Function (EF) is another evaluation matric that was proposed by Araújo *et al.* [65] to accurately checkthe validity of prediction. EF in Equation 9 is designed by combining MMRE and PRED to improve the performance of evaluation in estimation models.

The MRE is considered and claimed as biased by many studiesbecause of its asymmetric distribution. Since both MMRE and PRED are based on MRE, they are also categorized as biased performance measures [66], [67]. Mean Absolute Error (MAE) on the other side does not produce asymmetric distribution. It canbe calculated by Equation 8 and 10. However, Itcould not be used in its form as it is very difficult to interpret due to non-standardized residuals Shepperd and MacDonell [67] introduced Standard Accuracy (SA) as in Equation 11 (where Mean Absolute Residual($MARP_0$) shows the mean of random guessing for a large number of executions) which was later improved by Langdon *et al.* [68] that helps to estimates the effect size as shown in Equation12 (where Sp_0 indicates the sample standard deviation for the random guessing). SA tests the prediction model if it generates the understandable predictions, in the other case the prediction model is not stated as useful. SA quantifies the reliability of an estimation model. The

TABLE 1. 3-fold cross-validation for DABE.

	set		
	set1 Basic	set2 Training	set3 Testing
Fold1	set1	set2	set3
	set1	set3	set2
Fold2	set2	set1	set3
	set2	set3	set1
Fold3	set3	set2	set1
	set3	set1	set2

negative values of SA are unacceptable whereas zero indicates that the estimation model is less reliable. The estimated results produced by predictive models are verified by the Effect Sizeas it checks and compares the model effectiveness with random guessing. Effect Size (Δ) categorizes values in large (0.8), medium (0.5) and small (0.2). If the value is greater then or equal to 0.5 the results are considered as favorable [44], [67].

B. TRAINING STAGE OF BABE

The ensembled estimation model is designed based on the weight adjustment of features by incorporating the flavor of ABC in the similarity function of ABE. The 'effort' feature of the datasets is taken as the target feature or the dependent feature and the rest of the features are grouped as the independent feature for development effort estimation. In the training stage, the total set of projects is divided into three different sets such as basic project sets, training project sets, and testing project sets. The first two are utilized for model training purposes and the third step is left for model testing purpose or evaluation motives. Training and testing projects are compared with the basic projects to find suitable weights from training projects and to evaluate the estimation model accuracy through testing projects.

In the execution of this activity, a project from the training set is taken as the targeted project (which is removed from the total set) which is passed through the weighted similarity function such as Euclidian or Manhattan. The optimized weights generated (ranged from 0 to1) by the Employee Bees, Onlooker Bees, and Scout Bees, are assigned to the independent features from the similarity function. A comparison between the removed project and the basic project set is performed to identify the most similar or closest by analogy project.

The most similar project found through similarity function is then dealt with the solution function of ABE to predict the development effort using the closest analogy, mean, median or inverse weighted mean. The development effort calculated by this solution function is then evaluated by MRE. The flow of these events is recursively repeated while each estimation is performed for all the training projects.

The goal of all software estimation models is to reduce MMRE and increase PRED(0.25) which leads this study to adjust BABE for minimizing the value of MMRE. MRE values calculated from the earlier step are passed through the ABC part of the BABE model. The produced weights are

TABLE 2. Dataset employed from PROMISE Repository and their statistical information.

Dataset	Features	Projects count	Unit	Min	Max	Median	Mean
Desharnais	12	77	Hours	546	23,940	3647	5046
Nasa93	3	18	Months	5	138.3	26.5	49.47
Maxwell	27	62	Hours	583	63,694	5189.5	8223.2
Cocomo81	17	63	Months	6	11,400	98	686
China	18	499	Hours	26	54,620	1829	3921

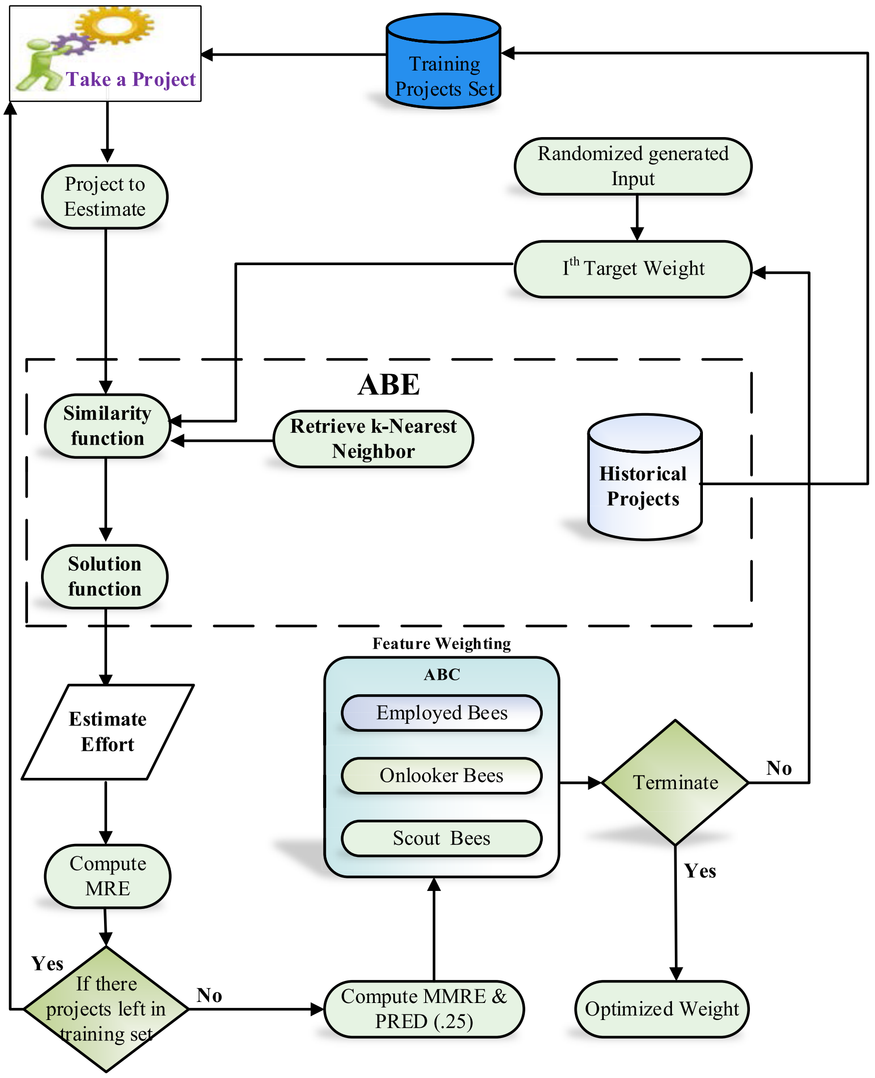


FIGURE 2. Training stage of BABE.

considered as the optimized weights if the stopping criterion is met. The optimized weights are infused in the similarity function to be used in the testing stage rest of the weights are reiterated in the training stage. Figure 2 depicts the complete procedure of the training stage of the BABE model.

C. TESTING STAGE

The testing stage uses training set projects to determine the performance of the BABE model's accuracy. The testing stage is almost similar to the training stage except for projects accompanied by the basic projects are from the testing set rather than the training set and it uses the generated optimized weights instead of producing these weights as done in the training stage. Further, the optimized weights produced in the training stage are utilized and infused in the similarity function for the purpose of model evaluation in the testing stage. Figure 3 shows the complete procedure of the training stage of the BABE model.

D. EVALUATION

The performance evaluation will be too much expectant if the accuracy is calculated based on the projects which are used during the model implementation. It may lead to a biased

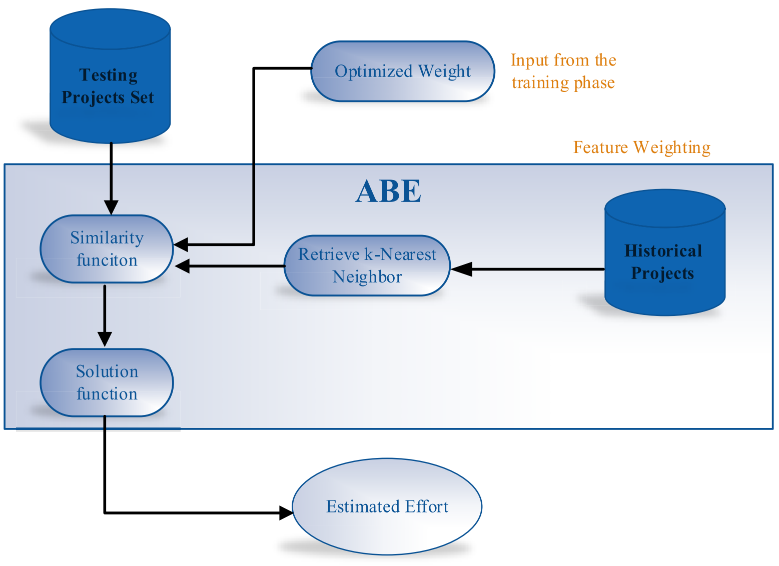


FIGURE 3. The Testing Stage of BABE.

model evaluation for estimation accuracy as the errors would always be low [34]. Therefore, a cross-validation method is used for pragmatic accuracy evaluation which divides the complete data into a number of train and test sets. The training sets contain the estimation results of datasets which are used during the model implementation. In the testing step, the estimation accuracies are evaluated using some unseen datasets. In the stage of testing, the estimation accuracies of all training sets and testing sets are combined for cross-validation. A three-fold cross-validation approached is adopted in this paper for evaluating the realistic accuracy of the model, as shown in the subsequent section.

1) CROSS-VALIDATION

There should be three main steps involved in the process of cross-validation, such as basic sets, training sets and testing sets. The basic, training and testing sets are randomly selected from the dataset which can be taken for n-fold cross-validation. 3-fold cross-validation should be preferred when the dataset is divided into three main groups. All data samples are partitioned into three groups on random bases, out of which two are grouped together in the training group/set, while the third sample is taken as the training set. These groups are formed three times to entertain all possible combinations.

In this study, a 3-fold kind of approach is adopted as shown in Table 1. There are six arrangements formulated in the proposed model as shown in Table 1, where 'set' reflects the basic group, which means the whole dataset. There are three subsets (set1, set2, set3) selected from 'set' as basic, train, and test sets. The number of projects in each set is the same. The performance is measured at each stage for two separate arrangements, the mean of which is considered as the resultant stage. The mean value calculated on the results of three stages determines the final results.

2) DATASET DESCRIPTION

There are six datasets used in this study to evaluate the performance of BABE. Five of these datasets are publicly available such as Desharnais, China, Maxwell, Nasa93, and Cocomo81 as shown in Table 2. The Desharnais dataset is based on Canadian projects. China dataset contains Chinese software projects. The Nasa93 and Cocomo81 datasets include Software projects from the United States. The Maxwell dataset is composed of Finnish banking projects. According to Dejaeger *et al.* [47] the datasets can be grouped into categories such as project data, development features, size features, environment features. Statistical information of six public datasets is shown in Table 2. Effort values of the datasets are unevenly distributed as determined by the skewness values of effort up to 6.6 [47], [48].

International Software Benchmarking Standard Group (ISBSG) is an Australia based organization, which collects information about software projects from around the world. The ISBSG Release 11 dataset is used in this study. This dataset contains information of 5052 projects. There are numerous attributes describing each software project of this dataset. The project attributes for this data are collected from 24 countries. The United States has made a major contribution with 31% of all the projects. Following the USA, Japan, Australia, and Finland are the three countries whose contribution to it is a double-figure of percentages such as 17%, 16%, and 10%.

VI. EXPERIMENTAL RESULTS

Estimation problems need data preprocessing before the model execution because the training quality can seriously be affected. This study normalizes all the independent features

Algorithm 1 Pseudo-Code of BABE

```
Algorithm: BABE
Input: $f$: Objective Function

Begin:
Step1:      Collect the data of previous projects to form a historical dataset.
Step2:      Select the project's appropriate features.
            $$Sim\left(p,p'\right)=\frac{1}{\left[\sqrt{\sum_{i=1}^{n}w_iDis\left(f_i,f_i'\right)}+\delta\right]}\delta=0.0001$$
            $$Dis\left(f_i,f_i'\right)=\begin{cases}\left|f_i-f_i'\right| & \text{if } f_i \text{ and } f_i' \text{ are numeric or ordinal}\\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i=f_i'\\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i\neq f_i'\end{cases}$$
Step2.1:    Find suitable weights for feature selection
Step2.1.1:  Initialize the food source
            $$x_{ij}=x\ min_j+rand(0,1)\ (x\ max_j-x\ min_j)$$
            where j represents the food source position from 1 to $D$ (Search Space Dimensionality), $x$ min $j$ and $x$ max $j$ shows
            the lower bound and upper bound of $j$; $i$ indicates the food source index from 1 to $SN$ (Population size)
Step2.1.2:  The employed bees treat their food source
            $$v_{ij}=x_{ij}+\varphi_{ij}(x_{ij}-x_{kj})$$
            where $j$ shows the randomly selected position, $x_i$ depicts the current food source, $x_k$ represents theselected food
            source for the current food source. $V_i$ is the treated food source by the $j^{th}$ parameter of $x_i$ and the randomly
            generated values ranging $-1$ to 1 is shown by $\varphi_{ij}$.
Step2.1.3:  The greedy selection is applied in $x_i$ and $v_i$., the employed bee memorizes $vi$ as the current sourceand leaves $x_i$ if
            $f(\upsilon_i)>f(x_i)$.
Step2.1.4:  A probability value is assigned to each food source
            $$p_i=\frac{fitness_i}{\sum_{i=1}^{SN}fitness_i}$$
            where $SN$ shows the size of population and fitness $_i$ depicts the value of food source $x_i$.
Step2.1.5:  The food source is selected by each of the onlooker bees in a probabilistic way and starts searching in a similar
            manner to the phase of employed bee
Step2.1.6:  A new source is generated by the scout bee using Step2.1.1 if the limit value determines any exhausted food source
Step2.1.7:  Step2.1.2 to Step2.1.6 are repeated until the stopping criterion is met (the maximum number of cycles)
END

Output: The Optimal Target Weight is Selected for the Testing Stage
```

ranging from 0 to 1 for producing the same effect on effort feature. PRED and MMRE are used to compare the accuracy of BABE model. There are 5 values ranging from 1 to 5 implemented to the solution function e.g. to the inverse weighted mean (Equation 3). The results produced by k Nearest Neighbor (kNN) are recorded, and so, to evaluate the effects of similarity function on the estimation process Euclidean similarity is used.

The results retrieved from the simulation procedure and the analysis performed on those results are discussed in this section. The simulation was conducted to retrieve the most appropriate adjustment for ABE through (k value, similarity measure, solution function, etc.) and the evaluation measures such as MMRE, PRED (0.25) and SA. Eventually, these performance measures were used to select the most accurate variant of ABE as a software development effort estimation model. BABE was found as the most appropriate model based on accurately estimated results. Desharnais and Maxwell datasets were taken to perform the experimental procedure. The generalization error for both the datasets can be seen in Table 3 and Table 4. For Euclidian Distance, K=3 was the most significant value as computed by MMRE. The K value at '5' produced the most significant configuration for ABE inverse weighted mean on both the datasets. Therefore, the Desharnais and Maxwell datasets were evaluated by SA too, to confirm the most significant configuration for ABE.

TABLE 3. Results of BABE on Desharnais Dataset.

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidean	1	Closest	0.014	0.676	0.051	0.891
	2	Inverse	0.058	0.125	0.091	0.193
		Mean	0.007	0.117	0.017	0.195
	3	Inverse	0.058	0.184	0.091	0.282
		Mean	0.039	0.127	0.019	0.291
		Median	0.061	0.118	0.020	0.298
	4	Inverse	0.058	0.242	0.091	0.379
		Mean	0.35	0.175	0.050	0.301
		Median	0.046	0.187	0.033	0.370
	5	Inverse	0.058	0.290	0.091	0.470
		Mean	0.054	0.294	0.082	0.483
		Median	0.079	0.242	0.057	0.441

TABLE 4. Results of BABE on Maxwell Dataset.

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidean	1	Closest	0.031	0.652	0.014	0.098
	2	Inverse	0.062	0.539	0.051	0.081
		Mean	0.091	0.049	0.041	0.079
	3	Inverse	0.062	0.089	0.051	0.981
		Mean	0.046	0.061	0.074	0.072
		Median	0.052	0.044	0.083	0.092
	4	Inverse	0.062	0.113	0.051	0.170
		Mean	0.059	0.079	0.300	0.176
		Median	0.040	0.069	0.605	0.148
	5	Inverse	0.062	0.147	0.051	0.161
		Mean	0.041	0.081	0.279	0.107
		Median	0.040	0.086	0.213	0.044

TABLE 5. SA results for Desharnais.

Similarity		Euclidean		
K		3	4	5
Training	Min. SA	11.263	10.156	12.102
	Max SA	92.362	90.831	42.124
	Ave. SA	39.652	35.197	27.415
	Std. SA	19.131	24.524	8.561
Testing	Min. SA	31.851	24.169	29.753
	Max SA	91.956	85.124	90.654
	Ave. SA	61.651	57.149	64.789
	Std. SA	22.165	15.324	12.003

The test results were evaluated against average, standard deviation, minimum and maximum values for SA. The SA results for Desharnais are shown in Table 5. The best values of SA as maximum and average fall at k=3, with average (39.652) and maximum (92.362) in the training stage. In the testing stage, the most suitable values occurred at k=3 with average (61.651) and maximum (91.956).

TABLE 6. Solution Function results for Desharnais for best k value.

Solution Function		Mean	Median	Inverse
Training	Min. SA	28.231	21.113	11.608
	Max SA	67.544	89.157	90.88
	Ave. SA	31.997	53.028	35.129
	Std. SA	3.891	18.712	34.568
Testing	Min. SA	31.124	48.251	27.627
	Max SA	86.528	81.065	84.205
	Ave. SA	54.981	64.501	81.957
	Std. SA	18.004	13.089	17.333

TABLE 7. SA results for Maxwell for best k value.

Similarity		Euclidean		
K		3	4	5
Training	Min. SA	13.354	14.312	17.245
	Max SA	97.542	92.148	31.246
	Ave. SA	51.495	39.131	24.261
	Std. SA	21.513	15.124	3.194
Testing	Min. SA	24.887	22.659	29.652
	Max SA	96.621	76.522	95.771
	Ave. SA	53.329	32.987	57.608
	Std. SA	10.362	6.99	9.035

SA values for the Maxwell dataset are shown in Table 7. The best SA values fall at k=3 with average (51.495) and maximum (97.542) in the training stage. In the testing stage, the most suitable values fall at k=3 with the average (53.329) and maximum (96.621). All the solution functions are not covered at k=1 and k=2 in the implementation,therefore, these values are not used in the comparison. The simulation was performed on SA for Desharnais and Maxwell datasets to further support the solution function against best k value. SA was selected as the benchmark for analyzing the results due to is the capability of generalization. Average, Standard Deviation, Minimum and Maximum of SA values of the solution functions (Inverse Weighted Mean, Mean and Median) for Desharnais dataset are shown in Table 6. The maximum and average SA occurred as 67.544 and 31.997 respectively. The Minimum, Maximum Average and Standard Deviation of SA values of the solution functions (Inverse Weighted Mean, Mean and Median) for Maxwell dataset can be seen in Table 7. The maximum and average SA occurred as 97.151 and 62.558 respectively. It should be noted that the results are generated from the selected projects, all the projects from data may affect the results and performance of our proposed solution.

VII. DISCUSSION

This section concludes that ES at k=3 for solution function is the most appropriate configuration for ABE based on the analysis performed in this section. The comparison and summary of the results for ABE, GAABE, DABE-3, PSO-ABE, and BABE can be seen in Table 9. The SA values for the BABE model on each dataset are as, China

TABLE 8. Solution Function results for Maxwell.

Solution Function		Mean	Median	Inverse
Training	**Min. SA**	28.124	41.257	13.001
	Max SA	85.524	91.133	86.358
	Ave. SA	46.682	73.008	35.82
	Std. SA	19.25	16.62	29.624
Testing	**Min. SA**	47.957	51.524	30.552
	Max SA	97.151	69.008	84.91
	Ave. SA	62.558	59.65	59.889
	Std. SA	17.657	10.916	15.701

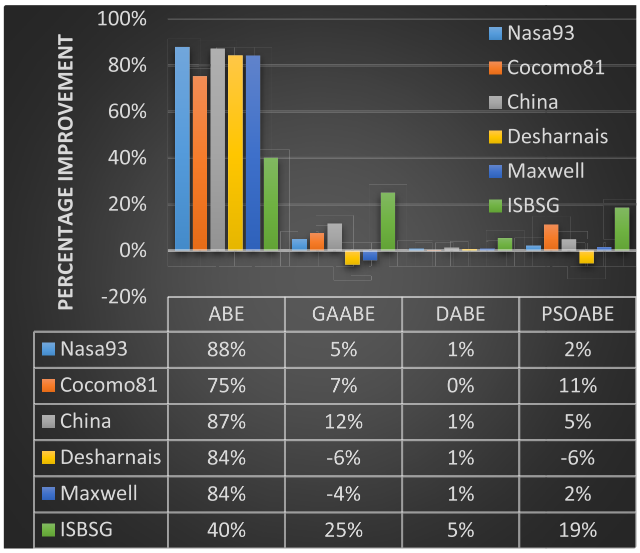
TABLE 9. Precision Values for Friedman Statistical Analysis.

Datasets	Estimation Models				
	ABE	**GAABE**	**DABE**	**PSOABE**	**BABE**
Nasa93	11.488	90.324	94.234	93.01	94.982
Cocomo81	24.531	91.812	98.94	88.016	99.201
China	12.493	86.196	96.509	92.88	97.621
Desharnais	13.235	89.332	83.66	88.854	84.205
Maxwell	13.411	88.423	84.18	83.63	84.91
ISBSG	41.27	51.638	65.09	56.08	68.82

(Training: 96.003, Testing: 97.62), Cocomo81 (Training: 96.821, Testing: 99.201), and Nasa93 (Training: 97.014, Testing: 96.180). The Δ values of this model for training and testing on China, Cocomo81 and Nasa93 are (0.228 and 0.213), (0.245 and 0.143) and (0.257 and 0.169) respectively. The detailed results analysis shows improvements in the estimation of BABE as compared to the existing models.

Figure 4 shows the percentage improvement of BABE against the existing models. It showed 2%, 1%, 5% and 88% improvement against PSOABE, DABE, GAABE, and ABE respectively on the Nasa93 dataset. On the Cocomo81 dataset, it showed 11%, 7% and 75% against PSOABE GAABE and simple ABE.

Its performance is found at par with DABE on the COCOMO81 dataset. On China dataset, it showed improvements of 5%, 1%, 12% and 87% against PSOABE, DABE, GAABE, and simple ABE respectively. It showed a percentage decrease of 6% against PSOABE and GAABE on the Desharnais dataset, whereas an improvement of 1% and 84 percent are found against DABE and simple ABE. On the Maxwell dataset, BABE showed 2%, 1% and 84% improvements against PSOABE, DABE, and simple ABE whereas it showed a 4% percentage decreased against GAABE. On the ISBSG dataset, which is the largest among all of the provided, it showed 19%, 5%, 25%, and 40% improvement against PSOABE, DABE, GAABE, and simple ABE respectively, which is quite a significant improvement.



	ABE	GAABE	DABE	PSOABE
Nasa93	88%	5%	1%	2%
Cocomo81	75%	7%	0%	11%
China	87%	12%	1%	5%
Desharnais	84%	-6%	1%	-6%
Maxwell	84%	-4%	1%	2%
ISBSG	40%	25%	5%	19%

FIGURE 4. Percentage improvement of BABE against the existing models.

The results revealed that the type and size of the dataset do affect the performance of weight optimization models for ABE. BABE showed significant performance on the ISBSG dataset which shows its supremacy on the existing weight optimization-based estimation models for ABE on the selected projects. Since the results on different datasets produced are different there has been a statistical analysis performed for validating the performance of the BABE model.

VIII. STATISTICAL PERFORMANCE EVALUATION

The software engineering datasets are heteroscedastic (there is a non-constant variance of subpopulations), therefore, non-parametric statistical tests are conducted for evaluating the estimation models' performance. Before conducting these non-parametric tests, the null hypothesis must be formulated which helps to evaluate the opposite situations of the alternate hypothesis. H_o represents the null hypothesis. This non-parametric test helps in statistical computation which rejects a hypothesis at a given significance (α) level. The smallest value of α concludes the null hypothesis rejection. This level indicates the p-value, which shows the probability of achieving at least as high as was expected conclusion while the null hypothesis is true. Instead of α, it is recommended to use p-value, because it can evidently estimate the result significance (the lower value of p shows the higher validation against the null hypothesis [69]. Pair wise and multiple comparisons are two major classifications of non-parametric tests. Multiple comparisons method is recommended to be used if there are more than two algorithms considered [70], [71]. To correlate the prediction models' results, various non-parametric tests can be performed. Friedman and Wilcoxon Signed tests belong to the two different classes of parametric tests. Friedman is used for analyzing multiple comparisons while the Wilcoxon Signed test conducts pair wise comparisons [70], [72]. The following null hypothesis is assumed.

TABLE 10. Friedman Test Statistics.

N	6
Chi-Square	15.733
df	4
Asymp. Sig.	.003

TABLE 11. Mean Ranks of Algorithms.

Algorithm/Model	Mean Rank
ABE	1.00
GAABE	3.17
DABE	3.50
PSOABE	2.83
BABE	4.50

TABLE 12. Descriptive Statistics of Friedman Test.

Algorithm		ABE	GAABE	DABE	PSOABE	BABE
N		6	6	6	6	6
Mean		19.4047	82.9542	87.1022	83.745	88.2898
Std. Deviation		11.73717	15.45686	12.52525	13.99284	11.47251
Minimum		11.49	51.64	65.09	56.08	68.82
Maximum		41.27	91.81	98.94	93.01	99.2
Percentiles	**25th**	12.2418	77.5565	79.0175	76.7425	80.3588
	50th (Medn)	13.323	88.8775	89.207	88.435	89.946
	75th	28.7158	90.696	97.1168	92.9125	98.016

$\mathbf{H_0}$**:** The existing imputation techniques used with ABE are equivalent to or better than the proposed ones

Multiple comparisons tests are performed for statistical analysis. For the Friedan test, at first, the original results are transformed into ranks which ranks each algorithm according to each dataset. The algorithm with the best values is assigned rank 1, the second-best is assigned rank 2 and so on. The Friedman test by Demšar [71] and García *et al.* [72] is utilized in this study for testing the null hypothesis. For the *Kth* prediction models and *N* datasets the ranks (r_i^j). The Friedman statistics as Equation 13 was obtained. The Chi-Square value is represented by x_f^2 in Equation 14.

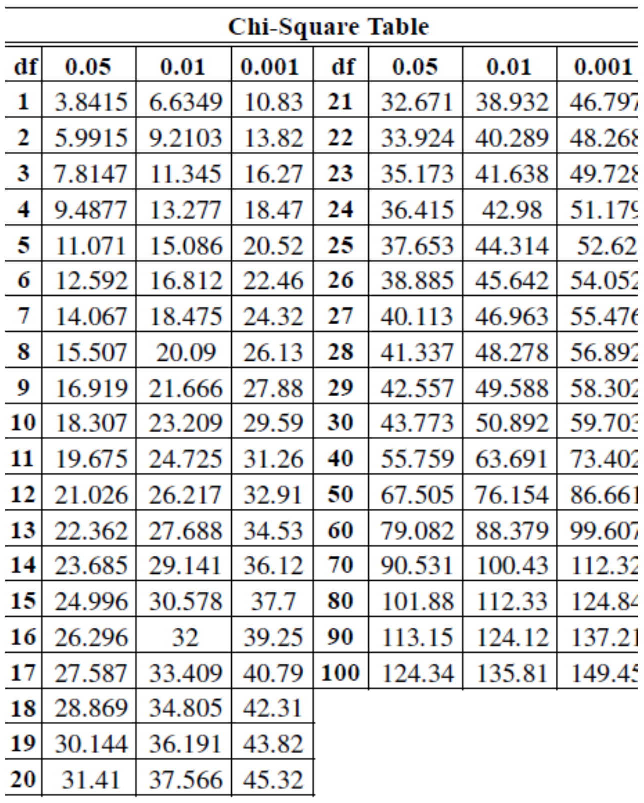
$$F_{F=}(N-1)x_F^2/N\ (K-1)-x_F^2 \tag{13}$$

where

$$x_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{14}$$

Table 13 (in appendix) provides the range of Chi-square values against the values of the Degree of Freedom (DF).

TABLE 13. Threshold values for rejecting H_0.



Chi-Square Table							
df	**0.05**	**0.01**	**0.001**	**df**	**0.05**	**0.01**	**0.001**
1	3.8415	6.6349	10.83	**21**	32.671	38.932	46.797
2	5.9915	9.2103	13.82	**22**	33.924	40.289	48.268
3	7.8147	11.345	16.27	**23**	35.173	41.638	49.728
4	9.4877	13.277	18.47	**24**	36.415	42.98	51.179
5	11.071	15.086	20.52	**25**	37.653	44.314	52.62
6	12.592	16.812	22.46	**26**	38.885	45.642	54.052
7	14.067	18.475	24.32	**27**	40.113	46.963	55.476
8	15.507	20.09	26.13	**28**	41.337	48.278	56.892
9	16.919	21.666	27.88	**29**	42.557	49.588	58.302
10	18.307	23.209	29.59	**30**	43.773	50.892	59.703
11	19.675	24.725	31.26	**40**	55.759	63.691	73.402
12	21.026	26.217	32.91	**50**	67.505	76.154	86.661
13	22.362	27.688	34.53	**60**	79.082	88.379	99.607
14	23.685	29.141	36.12	**70**	90.531	100.43	112.32
15	24.996	30.578	37.7	**80**	101.88	112.33	124.84
16	26.296	32	39.25	**90**	113.15	124.12	137.21
17	27.587	33.409	40.79	**100**	124.34	135.81	149.45
18	28.869	34.805	42.31				
19	30.144	36.191	43.82				
20	31.41	37.566	45.32				

DF is equal to K-1, therefore, in the experiments performed, the value of K = 5 and the value of DF = 4.

The related studies considered the sigma value of x_f^2 as 0.01 or less. According to the Chi-square table (Table 13) the x_f^2 the value should be greater than 13.277. The Friedman test statistics can be seen in Table 10. The Chi-square value is computed as 15.733 which shows that the null hypothesis is rejected. The test ranks of each model are shown in Table 11 and the descriptive statistics of the Friedman Test are shown in Table 12.

Once a null hypothesis is rejected, the second task is to identify the best and worst-performing algorithm. This information can be derived from the ranks in Table 11. According to Table 11, BABE is the best performing estimation model followed by DABE. ABE ranked lowest of the available models for comparison.

IX. CONCLUSION

Estimating the accurate software development effort estimation has been a challenge because of the complex and inconsistent nature of the software project. To estimate the cost of a targeted project, comparison with the past projects is performed. In this regard, ABE is the most widely adopted effort estimation model but even it is a widely used comparison-based estimation model, it still some time produces incorrect estimation results. The BABE model was produced in this study which ensembles ABC with ABE for feature weight optimization and accurate effort estimation by comparing the targeted project with the historical projects. The proposed model works in a two-stage environment based on the testing stage and training stage. In the training stage of BABE, the most appropriate weights are calculated which are then used in the testing stage for evaluating the estimation accuracy of the BABE model. There are six real datasets used in this study with the performance MMRE, PRED (0.25), SA and Δ performance metrics. Based on the SA and Δ values computed for this model, it can be concluded that the proposed model is more accurate than the existing development estimation model. In future works, it is intended to propose and combine missing data imputation techniques with the model in this study to strive for further improvement.

APPENDIX

See Table 13.

REFERENCES

[1] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empirical Softw. Eng.*, vol. 19, no. 4, pp. 857–884, Aug. 2014.

[2] C. Jones, *Estimating Software Costs: Bringing Realism to Estimating*. Osborne, Kansas: McGraw-Hill, 2007.

[3] B. W. Boehm and R. Valerdi, "Achievements and challenges in cocomo-based software resource estimation," *IEEE Softw.*, vol. 25, no. 5, pp. 74–83, Sep. 2008.

[4] B. Boehm, "Constructive cost model," in *Software Engineering Economics*. 1981.

[5] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Softw. Eng.*, vol. SE-9, no. 6, pp. 639–648, Nov. 1983.

[6] B. W. Boehm, R. Madachy, and B. Steece, *Software Cost Estimation With Cocomo II With Cdrom*. Upper Saddle River, NJ, USA: Prentice-Hall, 2000.

[7] N. Dalkey and O. Helmer, "An experimental application of the DELPHI method to the use of experts," *Manage. Sci.*, vol. 9, no. 3, pp. 458–467, Apr. 1963.

[8] K. Moløkken and M. Jørgensen, "Expert estimation of Web-development projects: Are software professionals in technical roles more optimistic than those in non-technical roles?" *Empirical Softw. Eng.*, vol. 10, no. 1, pp. 7–30, Jan. 2005.

[9] M. Jørgensen and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation," *J. Syst. Softw.*, vol. 83, no. 1, pp. 29–36, Jan. 2010.

[10] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, Nov. 1997.

[11] Y. F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," *J. Syst. Softw.*, vol. 82, no. 2, pp. 241–252, Feb. 2009.

[12] R. Bhatnagar, V. Bhattacharjee, and M. K. Ghose, "Software development effort estimation–neural network Vs. regression modeling approach," *Int. J. Eng. Sci. Technol.*, vol. 2, no. 7, pp. 2950–2956, 2010.

[13] M. A. Ahmed, M. Omolade Saliu, and J. AlGhamdi, "Adaptive fuzzy logic-based framework for software development effort prediction," *Inf. Softw. Technol.*, vol. 47, no. 1, pp. 31–48, Jan. 2005.

[14] M. Azzeh, D. Neagu, and P. I. Cowling, "Fuzzy grey relational analysis for software effort estimation," *Empirical Softw. Eng.*, vol. 15, no. 1, pp. 60–90, Feb. 2010.

[15] A. Idri, F. A. Amazal, and A. Abran, "Accuracy comparison of analogy-based software development effort estimation techniques," *Int. J. Intell. Syst.*, vol. 31, no. 2, pp. 128–152, Feb. 2016.

[16] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," *Swarm Evol. Comput.*, vol. 38, pp. 158–172, Feb. 2018.

[17] A. Khatibi Bardsiri and S. M. Hashemi, "A differential evolution-based model to estimate the software services development effort," *J. Softw., Evol. Process*, vol. 28, no. 1, pp. 57–77, Jan. 2016.

[18] Y. Cao, Y. Lu, X. Pan, and N. Sun, "An improved global best guided artificial bee colony algorithm for continuous optimization problems," *Cluster Comput.*, vol. 22, no. S2, pp. 3011–3019, Mar. 2019.

[19] L. Bao and J.-C. Zeng, "Comparison and analysis of the selection mechanism in the artificial bee colony algorithm," in *Proc. 9th Int. Conf. Hybrid Intell. Syst.*, 2009, pp. 411–416.

[20] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," *Empirical Softw. Eng.*, vol. 4, no. 2, pp. 135–158, 1999.

[21] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," *Empirical Softw. Eng.*, vol. 5, no. 1, pp. 35–68, 2000.

[22] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd, "Experiences using case-based reasoning to predict software project effort," in *Proc. EASE Conf.*, Keele, U.K., 2000, pp. 1–22.

[23] J. W. Keung and B. Kitchenham, "Optimising project feature weights for analogy-based software cost estimation using the mantel correlation," in *Proc. 14th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2007, pp. 222–229.

[24] J. Wen, S. Li, and L. Tang, "Improve analogy-based software effort estimation using principal components analysis and correlation weighting," in *Proc. 16th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Dec. 2009, pp. 179–186.

[25] Z. Pawlak, "Imprecise categories, approximations and rough sets," in *Rough Sets*. Springer, 1991, pp. 9–32.

[26] J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, "A flexible method for software effort estimation by analogy," *Empirical Softw. Eng.*, vol. 12, no. 1, pp. 65–106, Jan. 2007.

[27] J. Li and G. Ruhe, "Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+," *Empirical Softw. Eng.*, vol. 13, no. 1, pp. 63–96, Feb. 2008.

[28] J. Li and G. Ruhe, "Decision support analysis for software effort estimation by analogy," in *Proc. 3rd Int. Workshop Predictor Models Softw. Eng. (PROMISE, ICSE Workshops)*, May 2007, p. 6.

[29] D. Ju-Long, "Control problems of grey systems," *Syst. Control Lett.*, vol. 1, no. 5, pp. 288–294, Mar. 1982.

[30] C.-J. Hsu and C.-Y. Huang, "Comparison of weighted grey relational analysis for software effort estimation," *Softw. Qual. J.*, vol. 19, no. 1, pp. 165–200, Mar. 2011.

[31] Q. Song and M. Shepperd, "Predicting software project effort: A grey relational analysis based method," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7302–7316, Jun. 2011.

[32] M. Jørgensen, U. Indahl, and D. Sjøberg, "Software effort estimation by analogy and 'regression toward mean,'" *J. Syst. Softw.*, vol. 68, no. 3, pp. 253–262, 2003.

[33] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *J. Syst. Softw.*, vol. 80, no. 4, pp. 628–640, Apr. 2007.

[34] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "Increasing the accuracy of software development effort estimation using projects clustering," *IET Softw.*, vol. 6, no. 6, pp. 461–473, 2012.

[35] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Inf. Softw. Technol.*, vol. 48, no. 11, pp. 1034–1045, Nov. 2006.

[36] S. Kumari and S. Pushkar, "A genetic algorithm approach for multi-criteria project selection for analogy-based software cost estimation," in *Computational Intelligence in Data Mining*, vol. 3. Springer, 2015, pp. 13–24.

[37] A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1155–1166, Nov. 2010.

[38] J.-C. Lin and H.-Y. Tzeng, "Applying particle swarm optimization to estimate software effort by multiple factors software project clustering," in *Proc. Int. Comput. Symp. (ICS)*, Dec. 2010, pp. 1039–1044.

[39] A. F. Sheta, A. Ayesh, and D. Rine, "Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for NASA projects: A comparative study," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 6, pp. 365–373, 2010.

[40] C. V. M. K. Hari and P. V. G. D. Prasad Red, "A fine parameter tuning for COCOMO 81 software effort estimation using particle swarm optimization," *J. Softw. Eng.*, vol. 5, no. 1, pp. 38–48, Jan. 2011.

[41] D. Wu, J. Li, and Y. Liang, "Linear combination of multiple case-based reasoning with optimized weight for software effort estimation," *J. Supercomput.*, vol. 64, no. 3, pp. 898–918, Jun. 2013.

[42] V. Khatibi 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," *Softw. Qual. J.*, vol. 21, no. 3, pp. 501–526, Sep. 2013.

[43] Q. Liu, X. Chu, J. Xiao, and H. Zhu, "Optimizing non-orthogonal space distance using PSO in software cost estimation," in *Proc. IEEE 38th Annu. Comput. Softw. Appl. Conf.*, Jul. 2014, pp. 21–26.

[44] M. Azzeh, A. B. Nassif, S. Banitaan, and F. Almasalha, "Pareto efficient multi-objective optimization for local tuning of analogy-based estimation," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2241–2265, Nov. 2016.

[45] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Genetic programming for effort estimation: An analysis of the impact of different fitness functions," in *Proc. 2nd Int. Symp. Search Based Softw. Eng.*, Sep. 2010, pp. 89–98.

[46] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.

[47] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: A comparative study," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 375–397, Mar. 2012.

[48] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The promise repository of empirical software engineering data," Tech. Rep., Jun. 2012.

[49] T. K. Sharma and M. Pant, "Halton based initial distribution in artificial bee colony algorithm and its application in software effort estimation," in *Proc. 6th Int. Conf. Bio-Inspired Comput., Theories Appl. (BIC-TA)*, 2011, pp. 80–84.

[50] F. S. Gharehchopogh and Z. A. Dizaji, "A new approach in software cost estimation with hybrid of bee colony and chaos optimizations algorithms," *Magn. Res. Rep.*, vol. 2, pp. 1263–1271, Nov. 2014.

[51] T. T. Khuat and M. H. Le, "Applying teaching-learning to artificial bee colony for parameter optimization of software effort estimation model," *J. Eng. Sci. Technol.*, vol. 12, no. 5, pp. 1178–1190, 2017.

[52] F. S. Gharehchopogh, I. Maleki, A. Kamalinia, and H. M. Zadeh, "Artificial bee colony based constructive cost model for software cost estimation," *J. Sci. Res. Develop.*, vol. 1, no. 2, pp. 44–51, 2014.

[53] R. Y. Pratama, R. Sarno, and Sholiq, "Optimizing COCOMO II parameters using artificial bee colony method," in *Proc. 11th Int. Conf. Inf. Commun. Technol. Syst. (ICTS)*, Oct. 2017, pp. 125–130.

[54] P. S. Rao, K. K. Reddi, and R. U. Rani, "Optimization of neural network for software effort estimation," in *Proc. Int. Conf. Algorithms, Methodol., Models Appl. Emerg. Technol. (ICAMMAET)*, Feb. 2017, pp. 1–7.

[55] T. T. Khuat and M. H. Le, "Optimizing parameters of software effort estimation models using directed artificial bee colony algorithm," *Informatica*, vol. 40, no. 4, 2016.

[56] F. S. Gharehchopogh, I. Maleki, and A. Talebi, "Using hybrid model of artificial bee colony and genetic algorithms in software cost estimation," in *Proc. 9th Int. Conf. Appl. Inf. Commun. Technol. (AICT)*, Oct. 2015, pp. 102–106.

[57] M. Reisi, P. Moradi, and A. Abdollahpouri, "A feature weighting based artificial bee colony algorithm for data clustering," in *Proc. 8th Int. Conf. Inf. Knowl. Technol. (IKT)*, Sep. 2016, pp. 134–138.

[58] E. Hancer, B. Xue, M. Zhang, D. Karaboga, and B. Akay, "A multi-objective artificial bee colony approach to feature selection using fuzzy mutual information," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 2420–2427.

[59] G. Yavuz and D. Aydin, "Angle modulated artificial bee colony algorithms for feature selection," *Appl. Comput. Intell. Soft Comput.*, vol. 2016, pp. 1–6, Feb. 2016.

[60] R. J. Kuo, S. B. L. Huang, F. E. Zulvia, and T. W. Liao, "Artificial bee colony-based support vector machines with feature selection and parameter optimization for rule extraction," *Knowl. Inf. Syst.*, vol. 55, no. 1, pp. 253–274, Apr. 2018.

[61] C. Ozturk, E. Hancer, and D. Karaboga, "Dynamic clustering with improved binary artificial bee colony algorithm," *Appl. Soft Comput.*, vol. 28, pp. 69–80, Mar. 2015.

[62] Y. Wang, L. Feng, and J. Zhu, "Novel artificial bee colony based feature selection method for filtering redundant information," *Int. J. Speech Technol.*, vol. 48, no. 4, pp. 868–885, Apr. 2018.

[63] H. Wang, H. Yu, Q. Zhang, S. Cang, W. Liao, and F. Zhu, "Parameters optimization of classifier and feature selection based on improved artificial bee colony algorithm," in *Proc. Int. Conf. Adv. Mech. Syst. (ICAMechS)*, Nov. 2016, pp. 242–247.

[64] E. Hancer, B. Xue, M. Zhang, D. Karaboga, and B. Akay, "Pareto front feature selection based on artificial bee colony optimization," *Inf. Sci.*, vol. 422, pp. 462–479, Jan. 2018.

[65] R. D. A. Araújo, A. L. I. Oliveira, and S. Soares, "A shift-invariant morphological system for software development cost estimation," *Expert Syst. Appl.*, vol. 38, no. 4, pp. 4162–4168, Apr. 2011.

[66] I. Myrtveit and E. Stensrud, "Validity and reliability of evaluation procedures in comparative studies of effort prediction models," *Empirical Softw. Eng.*, vol. 17, nos. 1–2, pp. 23–33, Feb. 2012.

[67] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.

[68] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, MARP0," *Inf. Softw. Technol.*, vol. 73, pp. 16–18, May 2016.

[69] J. H. Zar, *Biostatistical Analysis*. New Delhi, India: Pearson Education, 1999.

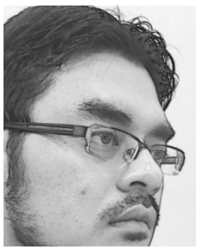
[70] J. Derrac, S. García, S. Hui, P. N. Suganthan, and F. Herrera, "Analyzing convergence performance of evolutionary algorithms: A statistical approach," *Inf. Sci.*, vol. 289, pp. 41–58, Dec. 2014.

[71] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[72] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power," *Inf. Sci.*, vol. 180, no. 10, pp. 2044–2064, May 2010.

MUHAMMAD ARIF SHAH graduated from the Department of Software Engineering, Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia. He is currently an Assistant Professor of Software Engineering with the Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur, Pakistan. He is also a member of the Software Engineering Research Group (SERG).

DAYANG NORHAYATI ABANG JAWAWI is currently an Associate Professor and the Deputy Dean (Academic and Student Development) of the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia. She is also a member of the Department of Software Engineering and the Software Engineering Research Group (SERG).



MOHD ADHAM ISA is currently a Senior Lecturer with the Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia.

MUHAMMAD YOUNAS graduated from the Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia. He is currently working as an Assistant Professor with the Department of Computer Science, Government College University Faisalabad.

ABDELZAHIR ABDELMABOUD is currently a Faculty Member with the Department of Information Systems, King Khalid University, Saudi Arabia.

FAUZI SHOLICHIN is currently a Research Student with the Department of Computer Science, Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Malaysia. He is also a member of the Software Engineering Research Group (SERG).