2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT) | 978-1-6654-5262-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCCNT54827.2022.9984558

Ensembling Teaching-Learning-Based Optimization Algorithm with Analogy-Based Estimation to Predict Software Development Effort

Pravali Manchala
Dept. of CSE
NIT Warangal
India
mpravali@student.nitw.ac.in

Manjubala Bisi
Dept. of CSE
NIT Warangal
India
manjubalabisi@nitw.ac.in

***Abstract*—Accurate evaluation of software development effort estimation is a significant challenge for project planning and management. The analogy-based estimation (ABE) model is an effective method to estimate the effort required for software development. In this study, we proposed a teaching-learning-based optimization (TLBO) guided analogy-based estimation (TABE) model in which TLBO is ensembled with ABE to estimate effort in a reliable way. TLBO generates optimized feature weights during training, and those optimized feature weights are used during testing to estimate development effort. Five datasets, including Kemerer, Desharnais, Albrecht, China and Maxwell, were used to validate the proposed TABE algorithm. The experimental findings demonstrate that TABE can provide a better PRED while minimizing the MMRE value for a few of the studied datasets.**

***Index Terms*—Analogy-based estimation, Software development effort, Teaching-learning-based optimization, Feature weight optimization.**

I. Introduction

The utilization of software constantly growing across all individual fields. It has become extremely important in the development, engineering, and acquisition of systems, especially for sophisticated and large-scale systems. For these kinds of systems, accurate development effort estimation is required and is regarded as a crucial component of efficient project management for software. It would be difficult to control and plan the project without accurate estimates [1]. Due to the rapid growth in the software industry, budget and time are the two most important factors to be considered. In a broad context, software effort estimation can be defined as a method of determining the required effort to develop or maintain software based on inadequate, uncertain, and/or noisy data input. Proper effort estimation is absolutely essential for feasibility studies, bidding, iteration plans, project planning, estimation of cost, and budgets for a project [2]–[4]. Furthermore, effort estimation can be useful in predicting future resource requirements.

Due to the nature of software projects, efforts can not be measured until work begins. Software estimation models have been far better than the traditional estimates used in the industry. One of the most popular estimation models for forecasting software development efforts is analogy-based estimation (ABE) [4], [5]. Researchers proposed various types of ABE model, but the most accurate methodology for accurate assessment could not be developed. While predicting, the regular classification models give equal priority to all features, but some of the features are not much related to effort and some are highly related to effort. Thus, numerous feature weight optimization techniques for similarity function in ABE have been proposed [4], [6]–[10], but no conclusion has been reached on the approach and environments that produce accurate estimates. Almost all algorithms require optimization of algorithm-specific parameters as well as some basic control parameters, making them difficult to use. The majority of algorithms require optimization of algorithm-specific parameters as well as some basic input variables, making them difficult to use and potentially degrade the prediction performance. So we chose an algorithm that doesn't demand such algorithm-specific parameters, which is the teaching-learning-based optimization (TLBO) [11]–[13] approach.

The major contributions in the paper are :

- Ensemble of Teaching-learning based optimization with Analogy-based estimation to design TABE model.
- Validation of the proposed TABE model with five datasets.
- Comparison of experimental results with some performance measures like PRED (Percentage of Prediction) (25) and MMRE (Mean Magnitude of Relative Error).

The paper is organized as follows: Related work is presented in Section II. The proposed approach is discussed in Section III. The details of experimental setup has given in Section IV, experimental results are discussed in Section V. Conclusion is drawn in Section VI.

II. Related Work

For the last few decades, researchers have focused on regression models for software effort estimation, while analogy-based estimations are giving better results than conventional regression models. Shepperd et al. [5] use anology-based

estimation and the ANGEL tool is used to automate the process. The model was evaluated on nine distinct industrial datasets with 275 projects, and it consistently outperforms stepwise regression-based algorithmic models.

Many studies have been focused on improving the software development effort, which is usually accomplished through feature weighting in ABE. It's a statistical strategy for determining how similar two projects are by comparing their features. It was chosen to increase the ABE performance because it also uses a comparison mechanism. To improve the performance of ABE, researchers have used Search-Based Software Engineering (SBSE) with advanced searching methods such as Artificial Bee Colony (ABC) [6], [14], Differential Evaluation (DE) [9], [15], Particle Swarm Optimization (PSO) [10], Simulated Annealing (SA), and Genetic Algorithm (GA) [7], [8].

GA is the most often used optimization technique in the ABE model for determining feature weights. Sun et al. [7] studied the use of a genetic algorithm to select the suitable weighted similarity measures of effort drivers in ABE models on estimation accuracy. They investigated three weighted analogy methods, namely linearly, unequally, and non-linearly weighted techniques. Li et al. [8] proposed feature weighing and project selection for ABE (FWPSABE) that limits the entire projects to a small subset consisting only of representative projects and determines optimized feature weights for ABE to further improve ABE estimation. GA was used as the FWPSABE system's optimization tool. On two artificial and two real-world datasets, they validated and compared them with traditional ABE, FWABE, and machine learning methods.

Benela et al. [9] assessed the effectiveness of ABE with the DE algorithm based on optimized feature weights. Five widely used mutation strategies are explored for optimizing the feature weights of ABE's similarity functions. Extensive simulation research was carried out on the PROMISE repository test suite and achieved substantial improvements in predictive performance against GA, PSO, and adaptive DE-based ABE models.

Amazal et al. [16] proposed a Fuzzy Analogy-based estimation model to predict development effort. Shah et al. [6] proposed an ABC with ABE (BABE) model to generate optimal weights in the ABE similarity function component. The extremely suitable weights are determined in the BABE training stage and are then used during the testing stage to evaluate the prediction accuracy of the BABE model. The estimation accuracy is evaluated using MMRE, PRED, and standard accuracy across five publicly available datasets and one ISBSG dataset.

Wu et al. [10] proposed a combination of six broadly used CBR (case-based reasoning) methods with PSO-determined optimized weights to estimate software effort. The experiments were conducted using the Miyazaki and Desharnais datasets, and the findings demonstrate that different CBR methods performed equally well for estimating software effort.

Classical optimization techniques are incapable of finding a global solution by uncovering irregular or non-linear surfaces. In order to overcome the limitations of the traditional strategy, a significant number of metaheuristic methods were investigated by researchers in a recent study to enhance the performance in terms of convergence and accuracy on complex problems [6], [7], [9], [10]. The most popular optimization techniques, such as GA, DE, and PSO, require a number of parameters to be tuned in their algorithms while performing solution optimization. Improper parameter value tuning results in the best solution being the local optimum.

TLBO is a robust method for solving the global optimization problem in which only population size and termination criteria need to be adjusted to obtain an optimal solution. TLBO can produce the global optimum solution at a lower cost and in less time.

III. Research Approach

A. Analogy Based Estimation

Shepperd et al. [5] presented ABE as a non-algorithmic estimate approach. It computes a new module's effort by comparing it to previous set of modules. ABE is usually divided into following four sections,

1) Historical data collection: to Create a historical dataset using the data of previously available projects.
2) Similarity measurement computation: to choose the appropriate features from projects.
3) Solution function computation: to find commonalities between the target project and data from previous initiatives. At this point, the weighted Euclidean and Manhattan distances are commonly selected.
4) Rule of association retrieval: to estimate the target project's software development effort.

1) Similarity Measure Computation: The similarity measure computation in ABE is utilized to compare the features of two projects. ABE compares target and previous project using Manhattan Similarity (MS) and Euclidean Similarity (ES). The below Eq 1 shows the ES between two projects.

$$Sim(q,q') = \frac{1}{\left|\sqrt{\sum_{i=1}^{m} w_i Dist(f_i, f_i')} + \delta\right|} \delta = 0.0001$$
$$Dist(f_i, f_i') = \begin{cases} (f_i - f_i') & \text{if } f_i \text{ and } f_i' \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases} \quad (1)$$

Although MS and ES have many similarities, MS estimates the absolute difference between characteristics. The below Eq. 2 shows MS between two projects,

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

From Eqs. 1 and 2, q and q' are the considered projects to compare, and the weight assigned to each feature represented by w_i, which range from 0 to 1. δ (δ=0.0001) is utilized to return a non-zero outcome. The project features are denoted by f_i and f_i', and the total count of features is represented by 'm'.

2) Solution Function: Once the K most comparable projects have been picked, the effort of the target project may be predicted or estimated based on the specified traits or features. The most popular solution functions [20,21,22] are the closest analogy, inverse weighted mean, mean and median of the most comparable projects. The median and inverse of effort can be computed from K $\geq$ 3 comparable projects, whereas the mean of effort can be computed from K $\geq$ 2 comparable projects. Equation 3 adjusts the part of each project by the inverse distance weighted mean in estimate.

$$C_q = \sum_{k=1}^{K} \frac{Sim(q, q_k)}{\sum_{i=1}^{n} Sim(q, q_i)} Cq_k \tag{3}$$

where the target project is represented by q, q_k represents k_{th} most similar project of q, Cq_k represents the effort value of q_k, the number of similar projects of target project (q) is denoted by K and 'n' represents total number of projects.

B. Teaching-Learning-based Optimization algorithm

The proper adjustment of algorithm-specific parameters in optimization techniques is a critical aspect that impacts the algorithms' performance. Incorrect tuning either increases processing effort or results in a local optimum solution. The Rao propsed teaching-learning-based Optimization (TLBO) algorithm doesn't require algorithm-specific parameters except population size [17]. This algorithm describes how learning can be done in two ways: (i) Teacher phase: learns from the teacher (ii) Learner Phase: by interaction with the other learner. In this algorithm, an environment is set up as a classroom with a population of students (learners), and different subjects are provided to them as features, with the student's result regarded as fitness value. The student with the greatest level of fitness is chosen as the teacher. The features of an optimization problem are the variables that influence the objective function, and the optimum solution is the ideal result of the objective function.

1) Teacher Phase: The learner with the best fitness value is considered as the teacher. During this phase, the teacher attempts to raise the mean of the population with respect to each feature. Assume there are 'm' subjects (i.e., features/attributes), 'n' learners (i.e., population size, i = 1, 2,..., n), and $[X_{Mean}]_{1\times m}$ is the mean of the learners with respect to all features (j = 1, 2,..., m). The greatest outcome among total population (X_{kbest}) obtained across all features in the overall population of learners can be regarded as the best learner 'kbest'. The difference between the current mean result for each feature and kbest learner is represented by Eq. 4:

$$Difference\_Mean = ran \times (X_{kbest} - T_F \times X_{Mean}) \tag{4}$$

where, X_{kbest} is the result of the best learner. T_F is the teaching factor which decides the value of mean to be changed, and ran is the random number in the range $[0, 1]$. The Value of T_F can be either 1 or 2 . The value of T_F is decided randomly with equal probability. The rest of the learners improve their fitness value using the above computed difference mean by Eq. 5:

$$X_i' = X_i + Difference\_Mean \tag{5}$$

Where X_i' is the new learned version of the current learner (X_i) from the population. In case X_i' achieves lower fitness (MMRE) value than the X_i, we update the learner otherwise keeps the current learner in population. All the learners following Eqs. 4 and 5 try to improve their fitness value from the best learner (teacher) in the teacher phase.

Algorithm 1 Pseudo-code of TABE approach

1: Divide dataset D in to training set and testing set.
2: **Training stage:**
3: Initialize randomly generated population for feature weights with size n×m.
4: Consider these weights to be the initial learners for the teaching learning based optimization.
5: ABE generated MMRE with these learners is considered to be the fitness function for the TLBO.
6: Calculate Initial fitness values for the initial learners using ABE model.
7: **for** i in 1 to n **do**
8: Calculate MMRE value with learner[i] as the feature weights for the ABE algorithm.
9: **end for**
10: **while** Termination criteria **do**
11: Update feature weights in population by minimizing error rate MMRE.
12: In teacher phase follows Eqs. 4 and 5 and update weights.
13: In learner phase follows Eqs. 6 and 7 and update weights.
14: **end while**
15: Return the optimal weights generated from training data at the end of TLBO iterations.
16: **Testing stage:**
17: On testing data, use optimized feature weights generated by training stage and finds similar projects through ABE's similarity function.
18: ABE's solution function uses those test data similar projects and estimates the effort.

2) Learner Phase: In the learner phase, each learner tries to learn from each other by sharing their knowledge.
Choose two learners A and B randomly A and B such that $X_A' \neq X_B'$ (where, X_A' and X_B' are the updated learners of X_A and X_B, respectively, at the end of teacher phase). There are two conditions to update learner X_A' in learning phase. As we considered minimization objective; we updated X_A' through Eq. 6 as long as the fitness value of X_A' is lesser than the X_B', or else updated X_A' through Eq. 7.

$$X_A'' = X_A' + ran \times (X_A' - X_B') \tag{6}$$

$$X_A'' = X_A' + ran \times (X_B' - X_A') \tag{7}$$

Where X''_A is the new learned version of the current learner (X'_A) from the population. In case X''_A achieves lower fitness (MMRE) value than the X'_A, we update the learner otherwise keeps the previously updated learner in population. All the learners following Eqs. 6 and 7 try to improve their fitness value compared to the other learners in the learning phase. The learning process iterates until termination criteria is satisfied and outputs the best feature weight set.

C. Proposed Teaching-Learning Based Optimization guided Analogy-Based Estimation (TABE) approach

The proposed TABE model basically consists of two stages, namely the training stage and the testing stage. In the training stage, training set data is used to optimize feature weights through the TLBO technique and estimate the training data effort using the ABE model. Later, in the testing stage, it utilizes the feature weights generated in the training stage to estimate testing data effort. The Algorithm 1 explains the detailed process of TABE, and the following are the concise steps of the TABE model:

- TLBO is utilized to generate optimum feature weights and uses ABE as an estimation model to find fitness value.
- ABE Choose a random target project from the training data and try to estimate its effort using the rest of the training data.
- It computes the K most similar projects to the target project in training data. While TLBO generated feature weights to be used in the similarity function.
- The solution function with similar projects estimates the effort of the target project and calculates the error in the estimated effort with the original effort and tries to minimize it.

1) Training Stage: Firstly, the TLBO algorithm attempts to find the optimal weights for the features in order to achieve the goal of a minimum MMRE on training data. In our proposed approach, TLBO considers the MMRE measure as a fitness function, and the TLBO's objective is to minimize the fitness function. We randomly initialized feature weights in the population of TLBO with a range from 0 to 1. To calculate MMRE (fitness function), TLBO calls the ABE model to find MMRE through estimated effort in the teacher phase and learner phase. The TLBO iterates on training data until termination criteria is met, at which point it returns the set of optimized best feature weights. These optimized weights are then used in the testing stage's similarity function.

Then in the ABE model, the dataset is divided into three sets- basic, training and testing sets. The "Effort" feature is regarded as the target feature, while the rest are regarded as independent features. Now, a project from a training set is removed and treated as a target project. The target project features are compared to projects in the basic set using the similarity function Manhattan or Euclidean as mentioned in the ABE Section III-A. The K most similar projects are chosen from the basic set by following the computation formulas given in Eqs: 1 and 2. The solution function of ABE then estimates the effort of the target project using the previously generated similar projects. The same process is followed for all the projects in the training set. The estimated effort is then compared to the actual effort value to determine MMRE and PRED values. TABE attempts to minimize MMRE while maximizing PRED values.

In the similarity function of ABE, every feature is assigned a weight, and similar projects may change based on the weight assigned to each feature. As a consequence, estimated effort changes according to the changes in weights of the features. So, by using the TLBO algorithm, we attempt to find the optimal weights for these features in order to achieve the goal of a minimum MMRE. The detailed step-by-step training stage operations are demonstrated in the Figure 1.

2) Testing Stage: After training the TABE model with training data, we tested the TABE model with testing data. The estimation accuracy of our model is determined at this stage. The testing stage is almost identical to the training phase, except the fact that we use the training stage's generated optimized weights whereas in the training stage it generates the weights. These optimized weights are then integrated into the similarity function of the ABE in the testing phase and compute the K most similar projects. The Solution function of the ABE model using similar projects of test data determines the software effort and computes the testing performance measures with respect to different K values. Figure 1 depicts the step-by-step testing operations.

IV. Experimental Setup

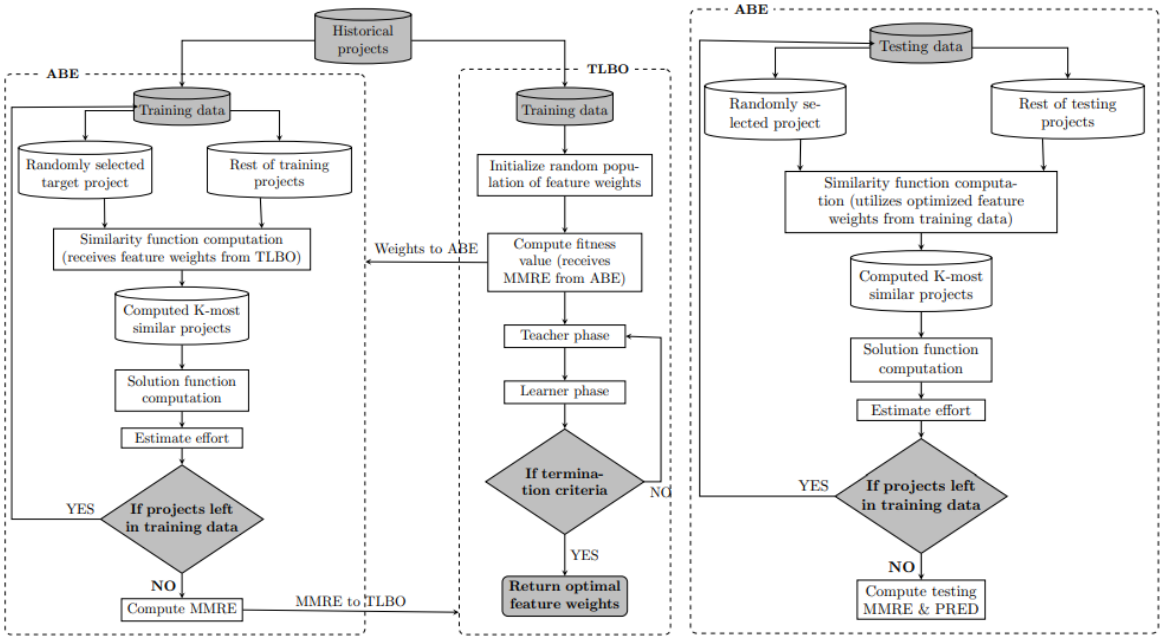
Evaluation of our model would be biassed if we used the same data for training as well as testing [6]. Hereby we employ a cross-validation technique, which divides the dataset into various training and testing sets. The training stage includes the training data estimations during model implementation. During the testing process, the prediction accuracy has been assessed using certain unknown datasets. This study employs a three-fold cross-validation (CV) technique to assess the model's realistic correctness, as indicated in the next subsection.

A. Cross-Validation

In our approach 3-fold CV divides the dataset into three sets (set1, set2, set3). These three sets are randomly selected and are of equal size. Two of these sets are merged to form a training set, and the third one is considered to be the testing set. To entertain all potential possibilities, these groupings are made three times. A 3-fold method is used and six arrangements are recommended in the proposed model. Basic, training, and testing sets are picked from the three subsets (set1, set2, set3). Each set has the same number of projects. The final outcomes are determined by the mean value obtained from the results of the three stages.

B. Experimental Objects

A total of five publicly available datasets are being used for the evaluation of our model, namely Albrecht, China,



(a) Training stage (b) Testing stage

Fig. 1: framework of a) training stage and b) testing stage in TABE model

Desharnais, Kemerer and Maxwell [6]. The Desharnais dataset is based on projects in Canada. Chinese software projects are included in China dataset. Maxwell is a collection of Finnish banking projects. The five public datasets information is given in Table I.

TABLE I: Description of datasets

Datasets	#features	#projects	Effort units
Kemerer	6	15	MM (man-month)
Desharnais	12	77	Hours
Albrecht	9	24	Hours
China	20	499	Hours
Maxwell	27	62	Hours

C. Performance Evaluation Measures

Numerous measures have been used to assess the performance of estimation methods in past research. We utilized the most prominent measures to assess our model, namely Percentage of Prediction (PRED), which serves as an additional criterion to calculate the proportion of estimates that are within less than 0.25 of the actual values and Mean Magnitude of Relative Error (MMRE), which estimates the average absolute percentage of difference between actual and expected effort [5], [6]. The computational equation of MMRE and PRED are shown in Eq. 9 and 10 simultaneously [10].

$$MRE = \frac{|Estimated\_effort - Actual\_effort|}{Actual\_effort} \quad (8)$$

$$MMRE = \sum \frac{MRE}{N} \quad (9)$$

$$PRED = \frac{A}{N} \quad (10)$$

In Eq. 9 and 10 , N represents the number of projects, A represents the number of projects with MRE $\geq$ 0.25. The ultimate focus of all the effort estimation models is to increase PRED while decrease MMRE.

V. Experimental Results

Estimation challenges necessarily require data transformation prior to model development because training performance can be adversely affected. Our work normalizes all of the attributes from 0 to 1 in order to produce the same influence on the effort feature. The accuracy of the TABE model is compared using PRED and MMRE. The outcomes of k Nearest Neighbor (KNN) are noted, and Euclidean similarity is used to assess the impact of the similarity function on the estimation method. This section discusses the simulation procedure's findings as well as the analysis performed on those results. The simulation work was executed to determine the most suitable ABE adjustments (k value, solution function, similarity measure, etc.) and performance metrics like MMRE and PRED (0.25). Finally, the most reliable adjustment of ABE as a software development effort estimation model was chosen using these performance measures. The TLBO has been performed around 150 iterations to get optimal feature weights. The MMRE and PRED value recorded with these optimal weights for training and testing sets are shown below.

The tables II-VI, represent the results obtained by Kermer, Desharnais, Albrescht, China and Maxwell datasets respectively. In each table, columns represent measures considered to find similarity, K value, solution functions for each K value, obtained MMRE and PRED results over training data and testing data. For all the experiments, we considered Euclidean

TABLE II: Results of TABE on Kemerer dataset

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidean	1	Closet	0.1401	0.3999	0.5086	0.6250
Euclidean	2	Inverse	0.1724	0.1333	0.3292	0.3999
Euclidean		Mean	0.2141	0.1000	0.3180	0.1750
Euclidean	3	Inverse	0.1388	0.2333	0.4156	0.3503
Euclidean		Mean	0.3128	0.3666	0.3034	0.3001
Euclidean		Median	0.2797	0.3000	0.2075	0.2750
Euclidean	4	Inverse	0.0964	0.3000	0.1413	0.5000
Euclidean		Mean	0.6081	0.4001	0.3638	0.2250
Euclidean		Median	0.3448	0.3333	0.2112	0.2250
Euclidean	5	Inverse	0.1215	0.2333	0.1944	0.5000
Euclidean		Mean	0.1796	0.1666	0.3198	0.1750
Euclidean		Median	0.2613	0.3333	0.2439	0.2750

TABLE III: Results of TABE on Desharnais dataset

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidian	1	Closest	0.3133	0.4552	0.1398	0.4924
Euclidian	2	Inverse	0.5332	0.1664	0.3201	0.5346
Euclidian		Mean	0.1451	0.3196	0.4194	0.1302
Euclidian	3	Inverse	0.1849	0.2650	0.4472	0.1116
Euclidian		Mean	0.5044	0.4061	0.2595	0.1802
Euclidian		Median	0.3070	0.1391	0.5606	0.4517
Euclidian	4	Inverse	0.4780	0.2127	0.5802	0.1681
Euclidian		Mean	0.2191	0.5495	0.4839	0.4221
Euclidian		Median	0.3738	0.2102	0.0896	0.3276
Euclidian	5	Inverse	0.3349	0.4137	0.5410	0.4829
Euclidian		Mean	0.0779	0.4179	0.4508	0.3667
Euclidian		Median	0.2257	0.2824	0.5872	0.1114

TABLE IV: Results of TABE on Albrecht dataset

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidian	1	Closest	0.3924	0.3834	0.3848	0.1619
Euclidian	2	Inverse	0.2760	0.1615	0.1739	0.2887
Euclidian		Mean	0.3318	0.3914	0.1879	0.3298
Euclidian	3	Inverse	0.3067	0.2318	0.2567	0.1069
Euclidian		Mean	0.3139	0.2773	0.1310	0.3838
Euclidian		Median	0.3796	0.2698	0.2994	0.2703
Euclidian	4	Inverse	0.1913	0.3018	0.2236	0.3510
Euclidian		Mean	0.3447	0.2639	0.1456	0.1887
Euclidian		Median	0.3844	0.1769	0.3634	0.2225
Euclidian	5	Inverse	0.1085	0.1591	0.1103	0.3055
Euclidian		Mean	0.1880	0.2170	0.1024	0.3789
Euclidian		Median	0.2951	0.2919	0.2321	0.3017

distance as the similarity measure. From these tables we can observe that, TABE on Kemerer dataset over training and testing data with k = 4 and inverse weighted mean solution gives significant MMRE value. The closet solution gives maximum PRED but at the cost of high MMRE on test data. TABE generalizes well on test data with K = 4 and an inverse weighted mean solution function by getting balanced values between MMRE and PRED. TABE on the Desharnais dataset generalizes well with K = 4 and a median solution function. TABE is unable to perform well over Desharnais. On the Albrecht dataset with a mean solution function, TABE gives the better MMRE and PRED than the other solution functions. K = 5 and K = 3 with mean solution function gives best on Albrecht. TABE over China dataset with K = 4, mean and inverse weighted mean solutions produces better MMRE and PRED values, except these other observations fail to achieves good outcome. TABE with the mean solution function gives good balancing results between MMRE and PRED on the Maxwell dataset. While other observations are unable to balance between MMRE and PRED values.

TABLE V: Results of TABE on China dataset

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidian	1	Closest	0.1234	0.0625	0.3101	0.1718
Euclidian	2	Inverse	0.1235	0.1458	0.3929	0.2187
Euclidian		Mean	0.3649	0.2083	0.3688	0.2343
Euclidian	3	Inverse	0.1136	0.2291	0.3115	0.2500
Euclidian		Mean	0.5421	0.3333	0.4568	0.4062
Euclidian		Median	0.5227	0.3125	0.3283	0.3906
Euclidian	4	Inverse	0.3187	0.3125	2.0898	0.3437
Euclidian		Mean	0.8044	0.4583	0.0475	0.4531
Euclidian		Median	0.2326	0.041	0.2608	0.1562
Euclidian	5	Inverse	0.2406	0.1875	0.7230	0.2656
Euclidian		Mean	0.1073	0.5416	0.7561	0.3281
Euclidian		Median	0.4233	0.3125	0.3558	0.2500

TABLE VI: Results of TABE on Maxwell dataset

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidian	1	Closest	0.0858	0.0370	0.2242	0.1759
Euclidian	2	Inverse	0.1577	0.1543	0.6249	0.3472
Euclidian		Mean	0.2777	0.2407	0.2755	0.1990
Euclidian	3	Inverse	0.3166	0.3209	0.2684	0.2453
Euclidian		Mean	0.1921	0.1852	0.2355	0.2870
Euclidian		Median	0.2022	0.1419	0.2537	0.2592
Euclidian	4	Inverse	0.1905	0.2098	0.4691	0.4259
Euclidian		Mean	0.1869	0.1851	0.2025	0.2500
Euclidian		Median	0.1846	0.1296	0.2160	0.1435
Euclidian	5	Inverse	0.1755	0.2098	0.3241	0.3194
Euclidian		Mean	0.2719	0.3148	0.3094	0.4188
Euclidian		Median	0.1909	0.1234	0.1987	0.1250

VI. Conclusion

Estimating the specific effort required for software development is a difficult task. Historical projects are examined to determine the cost of the target project. As a result, ABE has been always employed for this purpose. To predict development effort, we combined TLBO with the ABE algorithm. TLBO optimizes feature weights in this model during the training phase, and these optimized weights are then used in the testing stages for model evaluation. TABE provides better PRED with minimized MMRE values. We conducted experiments on only five publicly available datasets and two performance measures; we are unsure how our method will perform with other commercial datasets and measures. However, we provided a detailed procedure for implementing our method. For future studies, we will consider comparing TABE with some state-of-the-art effort estimation models and will consider performing statistical comparison analysis to validate the model.

References

[1] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, 2012.

[2] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on software engineering*, vol. 33, no. 1, pp. 33–53, 2006.
[3] P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger, "A probabilistic model for predicting software development effort," *IEEE Transactions on software engineering*, vol. 31, no. 7, pp. 615–624, 2005.
[4] M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl, "Optimal project feature weights in analogy-based cost estimation: Improvement and limitations," *IEEE Transactions on Software Engineering*, vol. 32, no. 2, pp. 83–92, 2006.
[5] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736–743, 1997.
[6] M. A. Shah, D. N. A. Jawawi, M. A. Isa, M. Younas, A. Abdelmaboud, and F. Sholichin, "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," *IEEE Access*, vol. 8, pp. 58 402–58 415, 2020.
[7] S.-J. Huang and N.-H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Information and software technology*, vol. 48, no. 11, pp. 1034–1045, 2006.
[8] Y.-F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," *Journal of systems and software*, vol. 82, no. 2, pp. 241–252, 2009.
[9] T. R. Benala and R. Mall, "Dabe: differential evolution in analogy-based software development effort estimation," *Swarm and Evolutionary Computation*, vol. 38, pp. 158–172, 2018.
[10] D. Wu, J. Li, and Y. Liang, "Linear combination of multiple case-based reasoning with optimized weight for software effort estimation," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 898–918, 2013.
[11] R. Venkata, *Teaching learning based optimization algorithm: and its engineering applications*. Springer International PU, 2016.
[12] R. V. Rao and V. D. Kalyankar, "Parameter optimization of modern machining processes using teaching–learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 524–531, 2013.
[13] J. Zhai, Y. Qin, Z. Zhao, and M. Yao, "Improved teaching-learning-based optimization algorithm," in *2018 37th Chinese Control Conference (CCC)*. IEEE, 2018, pp. 3112–3116.
[14] Y. Cao, Y. Lu, X. Pan, and N. Sun, "An improved global best guided artificial bee colony algorithm for continuous optimization problems," *Cluster computing*, vol. 22, no. 2, pp. 3011–3019, 2019.
[15] A. Khatibi Bardsiri and S. M. Hashemi, "A differential evolution-based model to estimate the software services development effort," *Journal of Software: Evolution and Process*, vol. 28, no. 1, pp. 57–77, 2016.
[16] F.-A. Amazal, A. Idri, and A. Abran, "An analogy-based approach to estimation of software development effort using categorical data," in *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*. IEEE, 2014, pp. 252–262.
[17] R. V. Rao, "Teaching-learning-based optimization algorithm," in *Teaching learning based optimization algorithm*. Springer, 2016, pp. 9–39.