Weight Optimization Based on Firefly Algorithm for Analogy-based Effort Estimation

Ayman Jalal AlMutlaq, Dayang N. A. Jawawi, Adila Firdaus Binti Arbain
Department of Software Engineering-School of Computing-Faculty of Engineering,
Universiti Teknologi Malaysia, Johor Bahru, Malaysia

***Abstract*—Proper cost estimation is one of the vital tasks that must be achieved for software project development. Owing to the complexity and uncertainties of the software development process, this task is ambiguous and difficult. Recently, analogy-based estimation (ABE) has become one of the popular approaches in this field due to its effectiveness and practicability in comparing completed projects and new projects in estimating the development effort. However, in spite of its many achievements, this method is not capable to guarantee accurate estimation confronting the complex relation between independent features and software effort. In such a case, the performance of the ABE can be improved by efficient feature weighting. This study introduces an enhanced software estimation method by integrating the firefly algorithm (FA) with the ABE method for improving software development effort estimation (SDEE). The proposed model can provide accurate identification of similar projects by optimising the performances of the similarity function in the estimation process in which the most relevant weights are assigned to project features for obtaining the more accurate estimates. A series of experiments were carried out using six real-world datasets. The results based on the statistical analysis showed that the integration of the FA and ABE significantly outperformed the existing analogy-based approaches especially for the ISBSG dataset.**

Keywords—Analogy-based estimation; firefly algorithm; software cost estimation; weight optimization

I. Introduction

Software development effort is considered one of the most significant measures estimated in the software projects owing to the fact that planning, developing, and all other vital processes of the project largely rely on correct estimation of the development effort [1]. Accurate estimation of software development metrics has become a critical issue for researchers in recent years in the software project management field [2-4]. The unstable nature of software project requirements, related hardware platforms, and the continuous change in software development frameworks complicate the process of estimation [5, 6]. Uncertain and insufficient available information to be used in equations, relations, formulas, and so on, become a major problem confronted by researchers in this field [4, 7].

Recently, analogy-based estimation has been found by many researchers as the most adaptable technique in software effort estimation [8, 9]. Analogy Based Estimation (ABE) can be defined as the selection of the previously completed projects similar in nature to the target project and deriving effort estimation based on these selected projects [10, 11]. Although the analogy-based estimation method is a simple and straightforward process, the process is extremely difficult due to the non-normality of software development data. [12]. Generally, the non-normality of software projects is the major issue that affects all comparison based approaches including the analogy based estimation method [13-15]. To address these issues thereby improving the estimation performance, appropriate weights of project attributes are evaluated in several research works [16, 17]. The weighting process is affected by irrelevant and complex projects and those projects that are out of the overall trend of the dataset [18, 19].

Various project attributes must be taken into consideration in the weighting process, compatible with principles of software engineering [20-22]. The inaccurate software development effort will result from attributes that are given the same weight even though they have different level of influence on estimation accuracy [23]. However, determining attribute weights used in the similarity function is a challenging issue in the ABE methods. Optimization, intensive search, and correlation analysis are the most prominent methods for attribute weighting. Correlation analysis tries to figure out the degree of dependency between software effort and other project attributes [24-26]. Intensive search applies in-depth search to determine the best subset of attributes [17, 27, 28]. Generally, the optimization methods tend to enhance the attribute weighting or feature selection in the ABE similarity function component [3, 29, 30].

Essentially, the majority of literature optimization approaches are motivated by nature, for example particle swarm optimization (PSO) which imitates fish schooling behaviour and bird flocking, ant Colony Optimization which imitates the ants' behaviour and the artificial Bee Colony (ABC) technique which mimics the bees' behaviour in searching for diet [31, 32]. Recently, the firefly algorithm (FA) which imitates some tropic firefly swarms has been introduced as a new metaheuristic algorithm [33]. Essentially, fireflies tend to be attracted to each other with higher intensity. This technique is typically different from other algorithms such as PSO and the Artificial Bee Colony (ABC). As such the FA can have two benefits: automatic regrouping and local attractions. As the intensity of light changes with distance, depending on the absorbing factor, the attraction between fireflies can be global or local, and therefore all global and local manners will be visited. Additionally, fireflies can also sub-divide and hence reorganize into sub-groups as neighbouring attraction is stronger than distant attraction; therefore it could be likely that each sub-group will group around a local mode [33-35]. This behaviour particularly helps the FA to be fit for the optimization problem.

Comparative studies revealed that the FA algorithm is very promising and could outperform many state-of-the-art optimization techniques like PSO and GA [36] , and Artificial Bee Colony ABC [37].Therefore, inspired by the above motivations among others, this research attempt to integrate FA with the ABE method to better optimize feature weights for improving the software development effort estimation. The main goal of this study is to improve the ABE model by optimizing the feature weights. To our knowledge, no research investigation has been conducted on the impact of FA on feature weighting for the ABE model.

Rest of the paper is organized as follow. Section II explains research background. The related work of the study is presented in Section III. The detail of the proposed work is described in Section IV. The experimental design is elaborated in Section V. Results and discussion of the study is detailed in Section VI. Section VII presents statistical analysis of the proposed model compared to related models. Section VIII concludes this research study.

II. Background

This section presents the background of the FABE model. We first discuss the concept of analogy-based estimation, which includes different steps of the ABE process. Further, each analogy estimation metric is described, which includes the similarity function and the solution function. Finally, in this section, the concept of the Firefly algorithm is also presented.

A. Analogy-Based Estimation (ABE)

The ABE method was initiated as a substitute for algorithmic-based software development effort estimation. In this technique, software project estimation is carried out by comparison with earlier accomplished projects and identifying the most similar projects to the board projects [38]. Owing to its suitability, the analogy-based estimation method has been popularly applied for software development in several studies. Essentially, ABE comprises four main modules, namely, historical dataset, K-nearest neighbours, similarity function, and solution function. More specifically, the ABE process is made up of steps as follows:

- Historical data creation through artificial or real datasets.
- Acquisition of new project features in a consistent manner with previous datasets.
- Applying predetermined similarity functions for example the Euclidean function to retrieve projects similar to the new projects.
- Predefined solution function is used to determine the new project' cost.

A similarity function is used in ABE for estimating the resemblance between two projects based on their feature comparison[38]. There are different similarity functions which include Manhattan similarity (MS) and the Euclidean similarity (ES). The Euclidean distance (ED) is the most popular similarity function which particularly involves distance between particular points. The similarity function is commonly used in optimization problems where distances are compared. MS is another popular similarity function in which the normal distance of Euclidean space is substituted by a new measurement where the distance between the locations is the sum of their coordinate's differences. These metrics are popularly applied for measuring the similarity in ABE. The nature of the projects at the normality level and the dataset can considerably affect the performance of similarity functions. ES function is shown in Equation 1:

$$Sim(p,p') = \frac{1}{\left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i f_i') + \delta}\right]} \quad (1)$$

$$Dis\ (f_i f_i') = \begin{cases} (f_i - f_i')^2 \ if\ f_i\ and\ f_i'\ are\ numerical\ or\ ordinal \\ 0\ \ if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i = f_i' \\ 1\ \ if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i \neq f_i' \end{cases} \quad (2)$$

Where, w_i is the weight (which ranges between 0 and 1), allocated to each feature, p, and p′ are the projects. f_i and f_i' represents the ith feature of each project, δ is used to gain a nonzero result and n represents the number of features.

The MS representation is like the ES formula, but it calculates the complete difference between features. The mathematical representation of the MS function can be given as:

$$Sim(p,p') \ = \frac{1}{\left[\sum_{i=1}^{n} w_i Dis(f_i\, f_i') + \delta\right]} \quad (3)$$

$$Dis\ (f_i f_i') = \begin{cases} |f_i - f_i'|\ if\ f_i\ and\ f_i'\ are\ numerical\ or\ ordinal \\ 0\ \ if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i = f_i' \\ 1\ \ if\ f_i\ and\ f_i'\ are\ nominal\ and\ f_i \neq f_i' \end{cases} \quad (4)$$

After identifying the K most similar projects, it would be possible to calculate the target project's effort based upon the selected features or attributes. The commonly used solution function include the Closest Analogy (CA) [11], the inverse weighted mean (IWM) [39], the average, and median of the most similar projects [40]. Mean is the average of effort for $K > 1$ while median is considered as effort median for similar projects with $K > 2$. In practice, Equation 5 adjusts the proportion of each project by using Inverse Weighted Mean (IWM).

$$C_p = \sum_{K=1}^{K} \frac{Sim(p,p')}{\sum_{i=1}^{n} Sim(p,p')} C_{p_k} \quad (5)$$

Where p, and p_k represents the new projects and the most similar kth project, respectively. C_{p_k} demonstrates the value of effort of $\boldsymbol{kth\ p_k}$ and $\boldsymbol{K}$ denotes the total number of the projects.

B. Firefly Algorithm

Yang developed the Firefly algorithm (FA) which reflects the characteristic flashing behaviour of fireflies [33]. Firefly algorithm comes with three assumptions: i) fireflies are unisexual: fireflies could attract each other irrespective of their gender. ii) The degree of attraction of fireflies is proportional to the brightness and both are inversely proportional to distance. If there are no brighter fireflies then fireflies will have random movement. iii) Firefly brightness is dependent on the objective function. In FA, fireflies show up in a swarm to resolve a particular optimization task through brightness which is identified by the fitness function, and movements of low brightness fireflies to high brightness which is determined by attractiveness.

In FA, the attraction between the flies involves two aspects; the various light intensities and the modeling of attraction. For a particular firefly at position X' brightness I is given as $I(X') \alpha f(x)$ while attraction β is proportional to the flies and is associated with the distance $R_{i,j}$ among fireflies i and j. Equation (6) demonstrates the inverse square of intensity $I(r)$ in which I_0 denotes the intensity of light from the source.

$$I(r) = I_0\, e^{-\gamma r^2} \tag{6}$$

Supposing an absorption factor of the environment γ, intensity is given in Equation 7 in which I_0 is the original intensity.

$$I(r) = \frac{I_0}{1+\gamma r^2} \tag{7}$$

Essentially, the ED is given in Equation 8, which signifies the distance between a firefly at position X_i and another at position X_j. Where X_{jk} is the k^{th} constituent of the spatial coordinate X_i

$$R_{i,j} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2} \tag{8}$$

A firefly i attracts a brighter one j as demonstrated in Eq .9 in which attraction can be given by $\beta_{e^{\gamma_{i,j}^2}} (x_i - x_j)$, and $\alpha \left[\text{rand} - \frac{1}{2}\right]$ denotes the randomness based on the randomization parameterα.

$$x_i = x_i + \beta_{e^{\gamma_{i,j}^2}} (x_i - x_j) + \alpha \left[\text{rand} - \frac{1}{2}\right] \tag{9}$$

Additionally, variations of attractiveness are controlled by γ which in turn influences the behavior and convergence speed of FA.

III. RELATED WORK

For the past years, several research works have been employed by different researchers to apply weighting techniques for improving ABE. One of these methods is using correlation coefficient analysis which is considered for feature selection and weighting in terms of software development effort estimation (SDEE) [41, 42] . In this case, project features with weak correlation are considered the low features and are assigned low weights while the features with higher correlation are given the higher weight and considered the most similar. The project features with no correlation are removed from the set of historical projects.

Weighting-based methods, known as Rough Set Analysis have been proposed for feature selection to better enhance the ABE performance [17, 43, 44]. In rough set analysis, feature dependency analysis generates several sub-sets of features named classes [45]. The most similar features are obtained by considering the intersection of all the classes. The frequency of attributes in reducts, the number of attributes in a core set, and the frequency of presence of attributes in decision rules are used to build the weighting model in the rough set technique. Another non-algorithmic method for estimation is Gray Theory (GT) in which gray depicts the fuzzy process, where the white and black represent known and unknown information respectively [45]. It is a statistical technique for finding the similarity degree by comparing two projects' features. Since it also uses a comparison technique, it was employed to enhance the ABE performances [46, 47]. One of the vital aspects of ABE is the solution function since it greatly influences the estimation performance's correctness. According to various studies, several attempts have been made to adjust expressions as the solution function to enhance performance [15, 48-50].

Over many years, to modify the feature weighting of the software estimation model, several optimization techniques have been introduced. The genetic algorithm (GA) is considered widely used optimization techniques for feature weights computation in the ABE. Huang and Chiu [51] utilized Genetic Algorithm to identify the best parameters in their defined non-linear/linear equation(s). The parameters involved in equations were determined as an improvement in the ABE’s performances. There has been a combination of various methods with a Genetic Algorithm for enhancing accuracy of estimation model such as the Gray Relational Similarity (GRS) method [46], regression techniques [52], and also linear adjustment [15]. For example, Bardsiri, et al. [12, 53] integrates Genetic Algorithms with fuzzy logic and artificial Neural Network, to develop a localized effort estimation process.

PSO has also been applied in many studies for improving the software development effort estimation. For example, Sheta, et al. [54, 55], Lin, and Tzeng [55] utilized the PSO technique to enhance the performances of the COCOMO estimation technique. In some scenarios, PSO has been shown to be more computationally efficient than GA [56]. Wu et al. applied the PSO algorithm for feature weight optimization in the predefined similarity measure of the software estimation approach [57]. Liu, et al. used PSO to reduce errors during the training phase and enhance estimation [58]. Azzeh, et al. [2] utilized the PSO algorithm to identify the optimum decision variable where the trade-off between several evaluation metrics is illustrated. Differential evolution have been used for feature weight optimization in ABE [23]. ABC has also been applied for the ABE optimization and indicated to outperform the PSO method [3]. Bardsiri, et al. integrated PSO with simulated annealing (SA) for feature weight optimization in ABE model [59]. Ferrucci, et al. [60] conducted a research on the influence of the fitness function. They showed that the model performance could be enhanced by choosing suitable and optimized performance measures. Essentially, the optimization of the fitness functions performs an important impact in estimation due to the complexity of software project.

IV. The Proposed FA-based Optimization for Analogy based Estimation (FABE)

In the proposed approach, the FA is integrated with the ABE model for improving the estimation accuracy. Adaptability and Flexibility are two important properties of the FA which make it capable to mitigate the issue of the vagueness and complexity of software project attributes [33, 61]. Essentially, the main purpose of the FA is to identify the most suitable feature weights that are to be used in the similarity function. Weights are allocated for parameter optimization to enhance the ABE performance. The system architectures of the training and testing of the proposed approach are illustrated in Fig. 1 and Fig. 2 respectively, whereas Algorithm 2 shows the Pseudo-code of FABE.

A. Training Stage

Fig. 1 illustrates the training phase architecture of the FABE approach. In the training stage, historical project data is utilized for predicting the efforts of the training dataset.

In this stage, the model adjusts the weights of features based on the FA in the Analogy-based Estimation similarity function. The dependent feature is the development effort; all others are considered independent features. In the training phase all available dataset projects are divided into (basic, train, test) subsets. For model construction in training stage basic and training subsets are used. For model evaluation in testing stage the basic and test subsets are involved. Training projects are compared with basic projects to find suitable weights and also testing projects are compared with basic for performance evaluation. A project is taken away from the training set and applied to the similarity function as a new project that is to be determined.

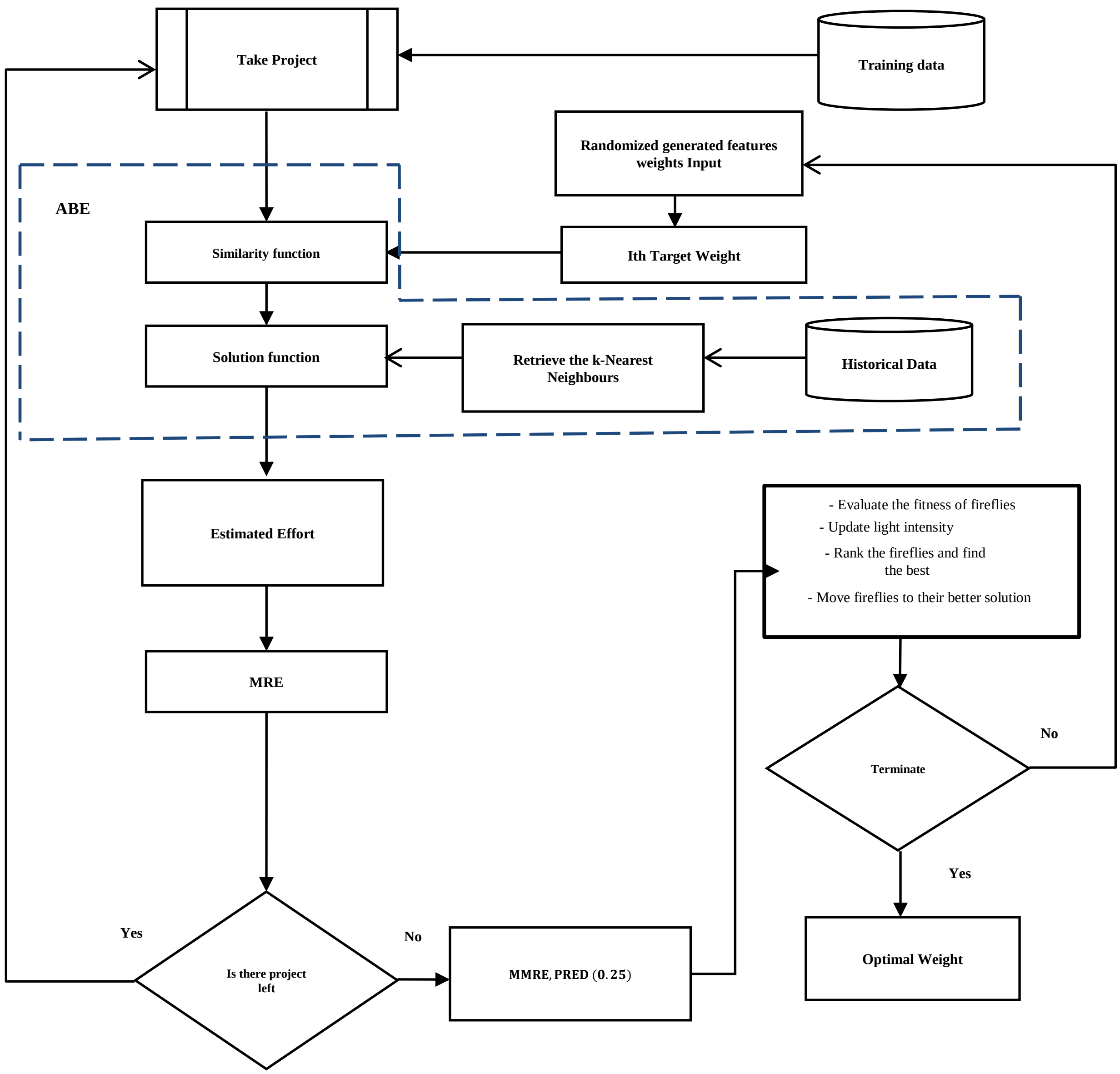

Fig. 1. Training stage.

Algorithm 1: FABE Algorithm

Inputs :

***f* (.)** objective function f(x)

□ randomized parameter

□□□ attraction coefficient

□ Light Absorption Coefficient

POP population Size

Step 1: Initialize the population of n fireflies.

Step 2: new project is selected form training dataset, and the remaining others projects are processed by ABE as historical projects.

Step 3: FABE feature weight parameter vector is encoded for the training project.

Step 4: weight vector of range [0, 1] is generated randomly.

Step 5: The training project similarity metric is evaluated for the weight vector selected randomly from pop.

Step 6: From the historical dataset obtain K closest analogies used in ABE, and then predict effort value of the training project using different solution functions.

Step 7: Until all training cases are treated with the same identical random weight vector (created for the first training case) repeat steps 2-6.

Step 8: MRE for each individual is calculated based on the objective function.

Step 9: Training projects set accuracy metrics (MMRE, PRED (0.25)) are evaluated.

Step 10: The Evolution step // Given that stopping criteria is not fulfilled.

Step 10.1: Evaluate the fitness of fireflies using objective function
Step 10.2: Update light intensity of the fireflies
Step 10.3: Rank the fireflies and find the best
Step 10.4: Move fireflies to their better solution
Step 10.5: Go to step 12 if the stopping criteria have been met. (maximum number of iterations)

Step 11: Go to Step 10.

Step 12: EXIT.

Output: The best Candidate solution with the optimal weight vector is chosen for the following Testing Stage.

The FA algorithm assigns weights to the independent features used in the similarity function. The considered project is compared with the basic projects based on the Equation (1). The most similar projects are discriminated by the similarity function to the removed project and take them to the solution function. Finally, the effort is determined in a solution function, and the MRE is calculated. This procedure is continued until all training projects are estimated. In the next step, the error and prediction performances MMRE and PRED (0.25) are calculated for a training group based on the MRE values. Reduce the value of MMRE and increases PRED (0.25) are the main goal of any estimation method which motivates this study to Fine-tune FA for MMRE value minimization.

The weights are recorded to be used in the testing stage if the termination requirements are met; otherwise, the FA updates the weights taking into account the obtained performance parameters. The similarity function is given new weights, and all computations are carried out once more for the training projects. Until the termination criteria are met, this process is continued. The training phase of the model is shown in Fig. 1. There are two rounds in the training phase, as shown in the image, with the first one having to do with calculating the MMRE for training projects and the second one having to do with adjusting weights using the FA approach.

B. *Testing Stage*

The primary purpose of this phase will be to assess the model's performance using hypothetical projects. Basic and testing projects are used as the inputs for the similarity function at this stage to examine how the suggested model performs. Additionally, the training stage's optimized weights are used to modify the similarity function. A project is separated from testing projects, as was done in the training stage, and then put up against basic projects through similarity function. Most resemble projects to the removed project are chosen and forwarded to predefined solution function. The amount of MRE is calculated after estimating the effort. This procedure is repetitive for all testing projects and, eventually, the value of MMRE and PRED are calculated. Fig. 2 illustrates the test phase of the proposed model. As previously mentioned, the project feature weights proposed by FA are produced to help the ABE accurately estimate the training projects effort as possible.

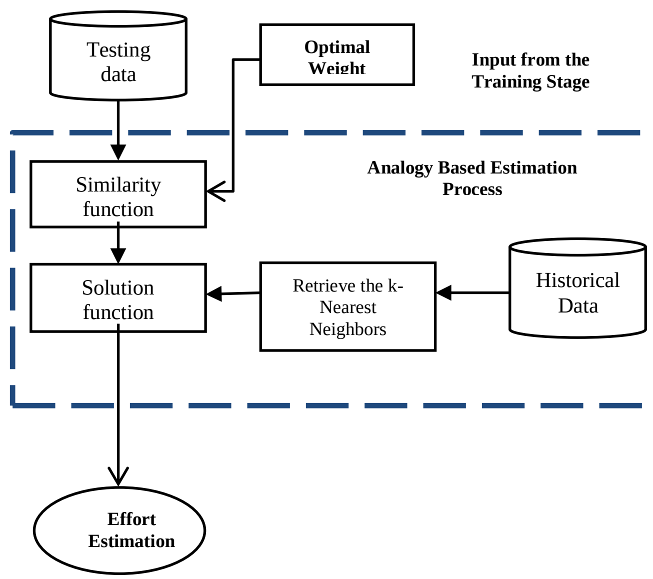

Fig. 2. Testing phase.

The ABE uses basic projects in this instance for comparison reason. Thus, two thirds of the current projects available in the given dataset (basic and training subsets) are utilized to obtain possible best weights, and the rest of the projects available are treated as a test set. To achieve the required precision, FA randomly produces the weights and modifies them over the iterations.

In the suggested approach, feature weighting is done thereby helping the ABE in producing better performances. The ideal set of feature weight vectors could differ from one execution run to the next because FA is a dynamic process that generates variable weights at each iteration. As a result, the weight allocated to a feature cannot be taken as the feature's importance but rather as a component that ABE must use.

V. EXPERIMENTAL DESIGN

The experimental strategy employed in this study is presented in the following section. First, the datasets used for the experiment to evaluate the accuracy of the proposed FABE technique are described, then the performance evaluation metrics. Further in this section, the process involved in the experimental setup, as well as the validation method, is explained and presented.

A. Dataset Description

To evaluate the performances of the proposed model, we employ six different datasets, namely, Desharnais, Maxwell, COMMMO, China, NASA, and International Software Benchmarking Standard Group (ISBSG) The Desharnais dataset contains Canadian projects. China dataset is based on Chinese software projects. United States software projects are contained in Cocomo81 and Nasa93 datasets. The Maxwell dataset is constructed based on of Finnish banking projects. Dejaeger et al. [62] claimed to group dataset to categories such as size, development, project data, and environment features. The Statistics of the datasets are given in Table I. Datasets effort values skewness is up to 6.6 [62, 63] Indicating asymmetrically distributed of effort for each dataset which is a thread for accurate estimation models.

International Software Benchmarking Standard Group (ISBSG) is established Australia software based non-commercial organization, which gathers data on software projects from numerous countries globally [64]. In this study ISBSG Release 11 dataset is employed. 5052 completed software projects detailed information have been collected. Software projects data collected from 24 countries are exist in the historical dataset. Majority of projects origin are come from USA with 30.7% percentage of all dataset, followed by Japan with 16.7%, Australia 15.9%, and Finland with 10.2%.

B. Cross Validation

The performance evaluation will typically be rather optimistic if the model accuracy is calculated based upon that projects that are used during the implementation phase.

As the errors will always be small, it might result in a biased model evaluation for estimating accuracy [56]. Thus, to better evaluate the model accuracy, a cross-validation approach is applied, which splits the entire dataset into several train and test sets. The results of datasets that are utilized during model construction are contained in the training sets. In testing stage, unseen datasets are utilized for evaluating the accuracy and the performance of all training and testing sets are merged for cross-validation. In this research, we used a three-fold cross-validation method for proposed model performance evaluation as illustrated in Table II.

As can be seen from Table II, six different arrangements can be considered for the model. Where S1, S2, and S3 are the three subsets randomly selected from the set of all projects as basic, training and testing sets accordingly. The sets involve the similar number of projects. At each fold, evaluation measure is calculated for two different arrangements, and the mean is considered as the result of that fold. Finally the accuracy is determined based on the mean of results computed from all three stages.

TABLE I. DATASETS USED IN EXPEREMENTS

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

TABLE II. ILASTRATION OF CROSS-VALIDATION

S			
S1	**S2**	**S3**	
Possible arrangements			
	Basic set	**Train set**	**Test set**
Fold 1	Set – 1	Set – 2	Set – 3
	Set – 1	Set – 3	Set – 2
Fold 2	Set – 2	Set – 1	Set – 3
	Set – 2	Set – 3	Set – 1
Fold 3	Set – 3	Set – 2	Set – 1
	Set – 3	Set – 1	Set – 2

C. Evaluation Metrics

Several metrics have been used by different studies for evaluating the performance of the comparison-based software estimation method. Accordingly, the most widely used measures include Relative Error (RE), Mean Relative Error (MRE), Percentage of Prediction (PRED) and Mean Magnitude of Relative Error (MMRE). The mathematical representation of these metrics can be given as follows:

$$\mathrm{RE} = \frac{\mathrm{estimated - Actual}}{\mathrm{Actual}} \tag{10}$$

$$\mathrm{MRE} = \frac{|\mathrm{Estimated - Actual}|}{\mathrm{Actual}} \tag{11}$$

$$\mathrm{MMRE} = \sum \frac{\mathrm{MRE}}{\mathrm{N}} \tag{12}$$

$$\mathrm{PRED\ (X)} = \frac{\mathrm{A}}{\mathrm{N}} \tag{13}$$

$$\mathrm{AE} = \mathrm{Estimated} - \mathrm{Actual} \tag{14}$$

$$\mathrm{EF} = \frac{\mathrm{PRED(25)}}{1 + \mathrm{MMRE}} \tag{15}$$

$$\mathrm{SA} = 1 - \frac{\mathrm{MAR}}{\mathrm{MAR}_{\mathrm{p}_0}} \tag{16}$$

$$\Delta = \frac{\mathrm{MAR} - \overline{\mathrm{MARp}_0}}{\mathrm{Sp}_0} \tag{17}$$

Projects with MRE >= X (X is usually reserved at 0.25) is denoted as PRED(X) as shown in Equation 6 and 7 where N denote the number of projects. Increase PRED and decrease MMRE is the main target of all software development effort estimation models, accordingly Araújo, et al. [65] proposed Evaluation Function (EF) measure that combined both MMRE and PRED in equation 9 to improve accuracy evaluation for software prediction model. MRE considered as a biased performance metric since its produce asymmetric distribution [22, 66]. MMRE and PRED both are derived from MRE they are also considered as biased performance measure.

Mean Absolute Error (MAE) produced non-asymmetric distribution on other side. Equation 8 and 9 shows how MAE can be calculated. Because of non-standardized residual MAE is difficult to interpret. SA was introduced by [22] it can be calculated by Equation 10 (in large number of runs the mean of random guessing is denoted as Mean Absolute Residual (MARpo)) which is enhanced by [67] and used later to estimates effect size as seen in Equation 11 (sample standard deviation for the random guessing is denoted as Spo). Reliability of the estimation model can be measured by SA as if the prediction model is stated as useful. SA negative values are not acceptable while zero value shows that the estimation model is unreliable. Effect size (Δ) evaluates the estimation results and the effectiveness of the mode is compared with random guessing. (Δ) categorizes values in small (0.2), medium (0.5) and large (0.8). If the value is equal or greater than to 0.5 the results is considered as favourable [2, 22].

VI. Experimental Results and Discussion

As stated earlier, the ABE technique uses three control parameters, including similarity function, solution function and K-nearest neighbour. In the experiment of this research, ED is adopted for similarity function. Median and mean, are considered as the solution function to compute the estimation values. This section presents the performance results of the proposed FABE model. We first present the experimental results in terms of the MMRE, MdMRE, and PRED based on the different control parameters, namely, Similarity, KNN, and solution function then the later the SA results. Further, the comparison results of the proposed model with existing methods are presented later in the section.

The proposed FABE model performance is compared and validated with commonly ABE weighting variants techniques, namely traditionally ABE, feature weighting with Genetic Algorithm in ABE (GAABE) [51] , feature weighting with Particle Swarm Optimization in ABE (PSOABE) [53], feature weighting with Differential evolution in ABE (DABE) [23], feature weight optimization with Bee colony optimization in ABE (BABE) [3], these estimation models are trained with historical data and algorithmic settings are tuned automatically.

A. Performance of the Proposed Model

Training quality estimation results are extremely affected by data pre-processing before main model execution started. In this study all the independent features were normalized in range (0 to 1) to produce same effect on software effort dependent feature. For the experimentation of the proposed FABE approach, we first investigate the possibility of getting the best settings of the model. To this end, we use different evaluation metrics namely (MMRE, MdMRE, PRED, and SA) on two different datasets which include Desharnais and Maxwell datasets. Also to assess the effect of the similarity function, the Euclidian similarity metric is employed. The results of the different values of the KNN (from 1 to 5) alongside respective solution function (Median, Inverse Weighted Mean, and Mean) metrics are recorded and shown in Table III to Table VI accordingly. Thus, in this section, the experimental results are presented and discussed. The main purpose of the experiment was to obtain the appropriate ABE configuration for the proposed model based on the different parameters (k value, similarity Metric, solution function).

Table III and Table IV demonstrate the simulation results of the FABE technique on the Desharnais and Maxwell datasets indicating various combinations of the key model parameters, such as KNN, similarity function, and solution function, respectively. From the results, it can be observed that the K value at 3 and Mean solution function are the most suitable setting as computed for both MMRE , MdMRE in the training and testing stage of the model on all the datasets, namely, Desharnais and Maxwell.

TABLE III. PERFORMANCE ON DESHARNAIS DATASET

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidean	1	Closest	0.015	0.685	0.056	0.889
	2	Inverse	0.051	0.127	0.089	0.201
		Mean	0.011	0.115	0.015	0.199
	3	Inverse	0.051	0.185	0.089	0.291
		Mean	0.033	0.131	0.017	0.299
		Median	0.059	0.115	0.021	0.289
	4	Inverse	0.051	0.245	0.089	0.381
		Mean	0.033	0.169	0.054	0.321
		Median	0.044	0.190	0.031	0.377
	5	Inverse	0.051	0.282	0.089	0.480
		Mean	0.049	0.245	0.081	0.488
		Median	0.081	0.245	0.055	0.452

TABLE IV. PERFORMANCE ON MAXWELL DATASET

Similarity	K	Solution	Training		Testing	
			MMRE	PRED	MMRE	PRED
Euclidean	1	Closest	0.041	0.701	0.019	0.095
	2	Inverse	0.059	0.542	0.049	0.091
		Mean	0.084	0.052	0.045	0.081
	3	Inverse	0.059	0.085	0.049	0.302
		Mean	0.044	0.059	0.070	0.069
		Median	0.061	0.042	0.081	0.089
	4	Inverse	0.059	0.117	0.049	0.181
		Mean	0.062	0.082	0.304	0.145
		Median	0.044	0.067	0.480	0.163
	5	Inverse	0.059	0.155	0.049	0.158
		Mean	0.040	0.077	0.271	0.104
		Median	0.039	0.086	0.220	0.126

However, the PRED (0.25) metric test performances showed that the most appropriate ABE setting for both these datasets was K=3 and the “Inverse Weighted Mean” solution function. Thus to further confirm the best configuration of the model thereby obtaining better performance, we also investigate evaluation for another performance measure SA on the Desharnais and Maxwell datasets.

Table V and Table VI demonstrate the SA results of the Euclidean similarity function for the K=3, K=4, and K=5 on the Desharnais and Maxwell datasets respectively concerning both testing and training stage of the proposed model. The results were evaluated against, maximum, minimum, average, and standard deviation for the SA. Table V demonstrates the SA values for Desharnais datasets. From results analysis, it can be concluded that best values of the SA as maximum and average in the training stage fall at K=3, with values of 95.120 and 39.899 respectively. Likewise based on testing stage, the more suitable values happened at K=3 with an average and maximum value of 63.436 and 93.688 respectively.

The SA results for the Maxwell datasets are demonstrated in Table VI. From observed results, it could be realized that the best performance in the training stage is at K=3 with the value of the Average SA and Maximum SA being 51.955 and 95.900 respectively. Similarly, based on testing stage, the most appropriate performance was observed at K=3 and the maximum and average values of 96.938 and 52.923 respectively. It should be noted that in the experiment, $K = 1$ and K=2 were not considered for comparison since at these values all solutions functions were not covered. Eventually, we also conducted a simulation study on SA for Maxwell and Desharnais datasets to father support investigation of solution function best suited for the ideal value of K. SA was chosen as a guideline principle for further results analysis in order to its generalization capability.

Table VII and Table VIII demonstrate the SA results on three solution functions, namely, Inverse Weighted Mean, Median, and Mean on the Desharnais and Maxwell datasets indicating the respective maximum, minimum, average, and standard deviation of the SA respectively. Table VII shows the Minimum, Minimum, Standard Deviation, and Average of the SA of solution function (median, mean, and inverse weighted mean) for the Desharnais dataset. From the results, it can be observed the average and maximum SA values were recorded to be 35.027 and 56.445 respectively for the “mean” as a solution function. Similarly, Table VIII shows lists of the maximum, minimum, average, and standard deviation of SA for the “mean”, median, and inverse solution functions for the Maxwell dataset. The results show that the average and optimal maximum SA values were reported to be 65.157 and 98.019, respectively, for the “mean” solution function. Based on the reported results in this section it was concluded that the most appropriate setting of the ABE is the “Mean “as a solution function and K=3.

B. Discussion

In this section the performance of the proposed FABE was validated and compared with the state-of–the-art ABE models, namely, traditional Analogy-Based Effort (ABE), GA-based ABE, PSO-ABE, and BABE (Bee Colony-based), and Differential Evolution in ABE(DABE). All estimation methods were adjusted automatically using historical datasets and the algorithm parameters.

TABLE V. SA RESULTS ON DESHARNAIS DATASET

K	Training				Testing			
	Max. SA	Min. SA	Avg.SA	Std.SA	Max. SA	Min. SA	Avg.SA	Std.SA
3	95.120	13.026	39.899	18.113	93.688	36.185	63.436	25.571
4	92.015	10.955	33.791	26.251	86.944	21.978	59.099	14.941
5	54.156	15.304	26.981	9.615	92.871	25.357	67.717	11.833

TABLE VI. SA RESULTS ON MAXWELL DATASET

K	Training				Testing			
	Max. SA	Min. SA	Avg.SA	Std.SA	Max. SA	Min. SA	Avg.SA	Std.SA
3	95.900	19.665	51.955	20.018	96.938	21.677	52.923	9.226
4	91.502	15.255	41.711	13.320	69.933	19.100	31.990	5.340
5	40.351	23.089	22.962	3.054	95.841	27.720	58.65	8.026

TABLE VII. RESULTS OF SOLUTION FUNCTIONS FOR BEST K VALUES ON DESHARNIAS DATASET

Solution Function	Training				Testing			
	Max. SA	Min. SA	Avg.SA	Std.SA	Max. SA	Min. SA	Avg.SA	Std.SA
Mean	56.445	29.843	35.027	3.018	89.825	32.421	56.198	17.220
IWM	89.801	32.086	36.412	37.06	86.522	28.599	71.759	19.001
Median	90.221	23.311	54.082	15.217	84.025	51.512	65.979	14.098

TABLE VIII. RESULTS OF SOLUTION FUNCTIONS FOR BEST K VALUES ON MAXWELL DATASET

Solution Function	Training				Testing			
	Max. SA	Min. SA	Avg.SA	Std.SA	Max. SA	Min. SA	Avg.SA	Std.SA
Mean	88.261	30.483	49.970	17.701	98.019	51.220	65.157	16.551
IWM	87.016	16.921	37.028	28.990	85.990	35.255	59.988	15.071
Median	91.910	44.526	71.990	18.007	72.051	54.044	58.100	13.166

TABLE IX. PRECISIONS VALUES FOR FRIEDMAN STATISTICAL ANALYSIS TEST

Estimation Models	Datasets					
	China	Cocomo81	Nasa93	Maxwell	Desharnais	ISBSG
ABE	12.493	24.531	11.488	13.411	13.235	41.27
GAABE	86.196	91.812	90.324	88.423	89.332	51.638
BABE	97.621	99.201	94.982	84.18	84.205	68.82
DABE	96.509	98.94	94.234	84.91	83.66	65.09
PSOABE	92.88	88.016	93.01	83.63	88.854	56.08
FABE	98.426	99.711	95.009	85.661	85.012	71.803

Table IX shows the SA comparison results of the proposed FABE model with the existing approaches on six datasets namely Desharnais, Maxwell, Nasa93, China, and ISBSG based on the "Mean" solution function and K=3 Euclidian similarity. The SA values of FABE model for training and testing on Cocomo81, Nasa93 and China are (97.005, 99.711), (96.752, 95.009) and (96.973, 98.426) respectively. The Δ values for this proposed model on each dataset are as, Cocomo81 (Training:0.234,Testing: 0.129), China (Training:0.219 ,Testing: 0.209) ,and Nasa93 (Training:0.251,Testing:0.159). From the detailed comparison results, it can be observed that the proposed FABE approach outperforms existing models.

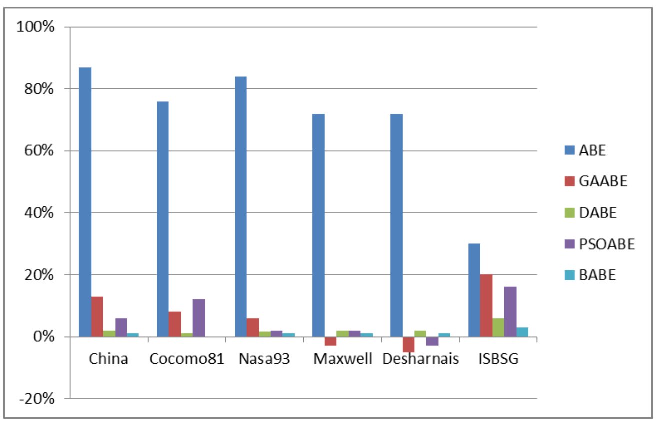

Fig. 3. FABE percentage of improvement against existing models.

TABLE X. FABE PERCENTAGE OF IMPROVEMENT

	ABE	GAABE	DABE	PSOABE	BABE
Cocomo81	76%	8%	1%	12%	0%
Maxwell	72%	-3%	2%	2%	1%
China	87%	13%	2%	6%	1%
Desharnais	72%	-5%	2%	-3%	1%
Nasa93	84%	6%	1.5%	2%	1%
ISBSG	30%	20%	6%	16%	3%

Fig. 3 and Table X demonstrate the average improvements achieved by the proposed FABE model compared to the existing models. For example in Cocomo81 dataset, it presented 8%, 1%, 12%, and 76 against GABE, DABE, PSOABE and traditional ABE. It presented 6%, 1.5%, 2%, 1% and 84% against GABE, DABE, PSOABE, BABE and traditional ABE respectively on Nasa93 dataset. For Cocomo81 dataset FABE performance is found at par BABE. In China, it presented improvements of 13%, 2%, 6%, 1% and 87% against GABE, DABE, PSOABE, BABE and traditional ABE respectively. It presented a percentage decrease of 5% and 3% against GABE and PSOABE respectively on desharnais dataset, whereas showed an improvement of 1%, 2%, and 72% against BABE, DABE and traditional ABE. In Maxwell dataset, FABE presented 2%, 2%, 1% and 72% improvement against DABE, PSOABE and traditional ABE respectively whereas it presented a percentage decrease of 3% against GABE. For ISBSG that considered the largest among all given datasets, it presented 20%, 6%, 16%, 3% and 30% against GABE, DABE, PSOABE,BABE and traditional ABE respectively, which is considered as significant improvement.

It can be concluded from result analysis that the size and type of dataset affect weight optimization techniques performance on ABE model. In the ISBSG dataset FABE outperformed existing optimization weight techniques significantly on the selected software projects for ABE model. Statistical analysis to validate FABE model performance is performed since results on different datasets are various.

VII. STATISTICAL PERFORMANCE EVALUATION

Statistical analysis is very important in finding the appropriateness of one technique to another. From the discussion in the previous section, it is obvious that the FABE approach provides the best results compared to the compared methods but now using statistical analysis this will be further confirmed. In this research owing to the fact that software engineering datasets have an issue such that each sub-population has non-contact variance, we employed nonparametric test for the analysis. A null hypothesis would be specified prior performing the test. This determines the differences or equality among the results of the models and enables alternative hypotheses to support the opposite condition to be assessed.

The null hypothesis is denoted as H_0, and the alternative hypothesis is represented as H_n. This test can be used to reject the hypothesis at a particular of significance level α. The p-value is indicated with this level, which represents achieve probability at least as high as expected while null hypothesis is valid. It is recommended to apply p-value instead of α since it can estimate results significantly (as p value is small this show strong validation against null hypothesis) [68]. Non-parametric tests can be classified into multiple comparisons like Friedman test and pair wise like Wilcoxon Signed test, in case of experiment that considers more than two algorithms or models it is recommended to use multiple comparisons test [69, 70]. In this case, the following hypothesis is considered:

H_0: All feature weight optimization prediction models are equivalent on ABE.

To test the null hypothesis, we employed the Freidman test that is stated by Demšar[70] and García, et al [71]. For Friedman test, initially, original results transformed into ranks each model according to each dataset. Best model value is assigned rank 1; second-best one is assigned rank 2 and so on. Accordingly, we assign ranks r_j^i , to the jth of k models on the ith of N data sets based on their accuracy. The Freidman statistic (F_f) can be given by equation.

$$F_f = \frac{(N-1)\chi_F^2}{N(K-1)-\chi_F^2} \quad (18)$$

Whereas is Chi-Square value is given by χ_F^2 in equation.

$$\chi_F^2 = \frac{12N}{K(K+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \quad (19)$$

The Degree of Freedom (DF) is equal to K-1, in this performed experiments, value of K=6 and so the value of DF=5. The sigma value of χ_F^2 in related studies is considered as 0.01 or less. Based on Chi-square table the value of χ_F^2 should be greater than 15.086. Friedman test statistic is presented in Table XII. Chi-Square value computed as 19.714, which allows the null hypothesis to be rejected. For each model test ranks are presented in Table XIII and also descriptive statistics of Friedman Test presented in Table XI.

The worst and best-performance model can be identified after the null hypothesis is rejected. From Table XIII which represents mean ranks best-worst performance information can be derived. It can be concluding from Table XIII that FABE is performing best model followed by BABE. The lowest ranked model among comparative models is ABE.

TABLE XI. DESCRIPTIVE STATISTIC OF FRIDMAN TEST

Model	N	Mean	Std. Deviation	Minimum	Maximum	Percentiles		
						25th	50th	75th
ABE	6	19.40467	11.737169	11.488	41.270	12.24175	13.32300	28.71575
GAABE	6	82.95417	15.456858	51.638	91.812	77.55650	88.87750	90.69600
DABE	6	87.10217	12.525246	65.090	98.940	79.01750	89.20700	97.11675
PSOABE	6	83.74500	13.992835	56.080	93.010	76.74250	88.43500	92.91250
BABE	6	88.28983	11.472514	68.820	99.201	80.35875	89.94600	98.01600
FABE	6	89.60367	10.847327	71.803	99.711	81.70975	91.33500	98.74725

TABLE XII. DESCRIPTIVE STATISTIC OF FRIEDMAN TEST

N	6
Chi-Square	19.714
df	5
Asymp. Sig.	.001

TABLE XIII. MEAN RANKS OF MODELS

Model	Mean Rank
ABE	1.00
GAABE	3.50
DABE	3.50
PSOABE	3.00
BABE	4.50
FABE	5.50

VIII. CONCLUSION

In this research, we proposed a weight optimization method for analogy-based estimation based on the firefly algorithm (FA). An estimation model is built and assessed during the training and testing phases of the suggested framework. FA considers all potential weights and chooses those that will produce the more accurate estimations. By giving project features the most suitable weights, the ABE method's comparison process quality was enhanced. Six datasets were used to test the accuracy of the proposed approach and a cross-validation method was used to calculate the performance metrics for the MMRE, PRED (0.25), MdMRE, SA, and Size Measure. The positive outcomes demonstrated that the suggested model can greatly improve the accuracy of estimations based on different metrics. The effectiveness of the proposed FABE technique was demonstrated in all datasets when the obtained results were contrasted with six widely used estimating models. The combination of FA and ABE resulted in a high-performance model for estimating software development effort, according to the findings from the datasets. In future work we intended to combine existing technique in this study with missing data imputation models to pursue for furthermore improvement on estimation accuracy.

REFERENCES

[1] Jones, T.C., Estimating software costs. 2007: McGraw-Hill, Inc.

[2] Azzeh, M., et al., Pareto efficient multi-objective optimization for local tuning of analogy-based estimation. Neural Computing and Applications, 2016. 27(8): p. 2241-2265.

[3] Shah, M.A., et al., Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction. IEEE Access, 2020. 8: p. 58402-58415.

[4] Gautam, S.S. and V. Singh, The state-of-the-art in software development effort estimation. Journal of Software: Evolution and Process, 2018. 30(12): p. e1983.

[5] Trendowicz, A. and R. Jeffery, Software project effort estimation. Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags, 2014: p. 277-293.

[6] Kaur, A. and K. Kaur, Systematic literature review of mobile application development and testing effort estimation. Journal of King Saud University-Computer and Information Sciences, 2022. 34(2): p. 1-15.

[7] Jadhav, A., M. Kaur, and F. Akter, Evolution of software development effort and cost estimation techniques: five decades study using automated text mining approach. Mathematical Problems in Engineering, 2022. 2022: p. 1-17.

[8] Wen, J., et al., Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology, 2012. 54(1): p. 41-59.

[9] Jorgensen, M. and M. Shepperd, A systematic review of software development cost estimation studies. IEEE Transactions on software engineering, 2006. 33(1): p. 33-53.

[10] Idri, A., F. azzahra Amazal, and A. Abran, Analogy-based software development effort estimation: A systematic mapping and review. Information and Software Technology, 2015. 58: p. 206-230.

[11] Walkerden, F. and R. Jeffery, An empirical study of analogy-based software effort estimation. Empirical software engineering, 1999. 4(2): p. 135-158.

[12] Bardsiri, V.K., et al., A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons. Empirical Software Engineering, 2014. 19(4): p. 857-884.

[13] Dolado, J.J., On the problem of the software cost function. Information and Software Technology, 2001. 43(1): p. 61-72.

[14] Li, Y.-F., 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, 2009. 82(2): p. 241-252.

[15] Chiu, N.-H. and S.-J. Huang, The adjusted analogy-based software effort estimation based on similarity distances. Journal of Systems and Software, 2007. 80(4): p. 628-640.

[16] Sigweni, B. and M. Shepperd. Feature weighting techniques for CBR in software effort estimation studies: a review and empirical evaluation. in Proceedings of the 10th International Conference on Predictive Models in Software Engineering. 2014.

[17] Li, J. and G. Ruhe, Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. Empirical Software Engineering, 2008. 13(1): p. 63-96.

[18] Sehra, S.K., et al., Research patterns and trends in software effort estimation. Information and Software Technology, 2017. 91: p. 1-21.

[19] Phannachitta, P. Robust comparison of similarity measures in analogy based software effort estimation. in 2017 11th International Conference on Software, Knowledge, Information Management and Applications (SKIMA). 2017. IEEE.

[20] Ali, A. and C. Gravino, Improving software effort estimation using bio-inspired algorithms to select relevant features: An empirical study. Science of Computer Programming, 2021. 205: p. 102621.

[21] Chen, Z., et al. Feature subset selection can improve software cost estimation accuracy. in ACM SIGSOFT Software Engineering Notes. 2005. ACM.

[22] Shepperd, M. and S. MacDonell, Evaluating prediction systems in software project estimation. Information and Software Technology, 2012. 54(8): p. 820-827.

[23] Benala, T.R. and R. Mall, DABE: Differential evolution in analogy-based software development effort estimation. Swarm and Evolutionary Computation, 2018. 38: p. 158-172.

[24] Keung, J.W., B.A. Kitchenham, and D.R. Jeffery, Analogy-X: providing statistical inference to analogy-based software cost estimation. IEEE Transactions on Software Engineering, 2008. 34(4): p. 471-484.

[25] Kocaguneli, E., et al., Exploiting the essential assumptions of analogy-based effort estimation. IEEE Transactions on Software Engineering, 2011. 38(2): p. 425-438.

[26] Malathi, S. and S. Sridhar, Detection of Aberrant Data Points for an effective Effort Estimation using an Enhanced Algorithm with Adaptive Features. Journal of Computer Science, 2012. 8(2): p. 195.

[27] Tosun, A., B. Turhan, and A.B. Bener, Feature weighting heuristics for analogy-based effort estimation models. Expert Systems with Applications, 2009. 36(7): p. 10325-10333.

[28] Jodpimai, P., P. Sophatsathit, and C. Lursinsap. Estimating software effort with minimum features using neural functional approximation. in 2010 International Conference on Computational Science and Its Applications. 2010. IEEE.

[29] Shahpar, Z., V. Khatibi, and A. Khatibi Bardsiri, Hybrid PSO-SA approach for feature weighting in analogy-based software project effort estimation. Journal of AI and Data Mining, 2021. 9(3): p. 329-340.

[30] Dashti, M., et al., LEMABE: a novel framework to improve analogy-based software cost estimation using learnable evolution model. PeerJ Computer Science, 2022. 7: p. e800.

[31] Slowik, A., Swarm Intelligence Algorithms: A Tutorial. 2020.

[32] Blum, C. and D. Merkle, Swarm intelligence: introduction and applications. 2008: Springer Science & Business Media.

[33] Yang, X.-S. Firefly algorithms for multimodal optimization. in International symposium on stochastic algorithms. 2009. Springer.

[34] Das, S., et al., Real-parameter evolutionary multimodal optimization—A survey of the state-of-the-art. Swarm and Evolutionary Computation, 2011. 1(2): p. 71-88.

[35] Fister, I., et al., A comprehensive review of firefly algorithms. Swarm and Evolutionary Computation, 2013. 13: p. 34-46.

[36] Yang, X.-S., Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409, 2010.

[37] Khaze, S.R., S. Hojjatkhah, and A. Bagherinia, Evaluation the efficiency of artificial bee colony and the firefly algorithm in solving the continuous optimization problem. arXiv preprint arXiv:1310.7961, 2013.

[38] Shepperd, M. and C. Schofield, Estimating software project effort using analogies. IEEE Transactions on software engineering, 1997. 23(11): p. 736-743.

[39] Kadoda, G., et al. Experiences using case-based reasoning to predict software project effort. in Proceedings of the EASE 2000 conference, Keele, UK. 2000. Citeseer.

[40] Angelis, L. and I. Stamelos, A simulation tool for efficient analogy based cost estimation. Empirical software engineering, 2000. 5(1): p. 35-68.

[41] Keung, J.W. and B. Kitchenham. Optimising project feature weights for analogy-based software cost estimation using the mantel correlation. in 14th Asia-Pacific Software Engineering Conference (APSEC'07). 2007. IEEE.

[42] Wen, J., S. Li, and L. Tang. Improve analogy-based software effort estimation using principal components analysis and correlation weighting. in 2009 16th Asia-Pacific Software Engineering Conference. 2009. IEEE.

[43] Li, J., et al., A flexible method for software effort estimation by analogy. Empirical Software Engineering, 2007. 12(1): p. 65-106.

[44] Li, J. and G. Ruhe. Decision support analysis for software effort estimation by analogy. in Third International Workshop on Predictor Models in Software Engineering (PROMISE'07: ICSE Workshops 2007). 2007. IEEE.

[45] Pawlak, Z., Rough Sets: Theoretical Aspects of Reasoning about Data Kluwer Academic Publishers. Dordrecht, 1991.

[46] Hsu, C.-J. and C.-Y. Huang, Comparison of weighted grey relational analysis for software effort estimation. Software Quality Journal, 2011. 19(1): p. 165-200.

[47] Song, Q. and M. Shepperd, Predicting software project effort: A grey relational analysis based method. Expert Systems with Applications, 2011. 38(6): p. 7302-7316.

[48] Jørgensen, M., U. Indahl, and D. Sjøberg, Software effort estimation by analogy and "regression toward the mean". Journal of Systems and Software, 2003. 68(3): p. 253-262.

[49] Kaushik, A., P. Kaur, and N. Choudhary, Stacking regularization in analogy-based software effort estimation. Soft Computing, 2022. 26(3): p. 1197-1216.

[50] Azzeh, M., A.B. Nassif, and L.L. Minku, An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. Journal of Systems and Software, 2015. 103: p. 36-52.

[51] Huang, S.-J. and N.-H. Chiu, Optimization of analogy weights by genetic algorithm for software effort estimation. Information and software technology, 2006. 48(11): p. 1034-1045.

[52] Oliveira, A.L., et al., GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. information and Software Technology, 2010. 52(11): p. 1155-1166.

[53] Bardsiri, V.K., et al., A PSO-based model to increase the accuracy of software development effort estimation. Software Quality Journal, 2013. 21(3): p. 501-526.

[54] Sheta, A.F., A. Ayesh, and D. Rine, Evaluating software cost estimation models using particle swarm optimisation and fuzzy logic for NASA projects: a comparative study. International Journal of Bio-Inspired Computation, 2010. 2(6): p. 365-373.

[55] Lin, J.-C. and H.-Y. Tzeng. Applying particle swarm optimization to estimate software effort by multiple factors software project clustering. in 2010 International Computer Symposium (ICS2010). 2010. IEEE.

[56] Bardsiri, V.K., et al., Increasing the accuracy of software development effort estimation using projects clustering. IET software, 2012. 6(6): p. 461-473.

[57] Wu, D., J. Li, and Y. Liang, Linear combination of multiple case-based reasoning with optimized weight for software effort estimation. The Journal of Supercomputing, 2013. 64(3): p. 898-918.

[58] Liu, Q., et al. Optimizing non-orthogonal space distance using pso in software cost estimation. in 2014 IEEE 38th Annual Computer Software and Applications Conference. 2014. IEEE.

[59] Shahpar, Z., V.K. Bardsiri, and A.K. Bardsiri, Polynomial analogy-based software development effort estimation using combined particle swarm optimization and simulated annealing. Concurrency and Computation: Practice and Experience, 2021. 33(20): p. e6358.

[60] Ferrucci, F., et al. Genetic programming for effort estimation: an analysis of the impact of different fitness functions. in 2nd International Symposium on Search Based Software Engineering. 2010. IEEE.

[61] Ghatasheh, N., et al., Optimizing software effort estimation models using firefly algorithm. arXiv preprint arXiv:1903.02079, 2019.

[62] Dejaeger, K., et al., Data mining techniques for software effort estimation: a comparative study. IEEE transactions on software engineering, 2011. 38(2): p. 375-397.

[63] Menzies, T., et al., The promise repository of empirical software engineering data. West Virginia University, Department of Computer Science, 2012.

[64] González-Ladrón-de-Guevara, F., M. Fernández-Diego, and C. Lokan, The usage of ISBSG data fields in software effort estimation: A systematic mapping study. Journal of Systems and Software, 2016. 113: p. 188-215.

[65] Araújo, R.d.A., A.L. Oliveira, and S. Soares, A shift-invariant morphological system for software development cost estimation. Expert Systems with Applications, 2011. 38(4): p. 4162-4168.

[66] Myrtveit, I. and E. Stensrud, Validity and reliability of evaluation procedures in comparative studies of effort prediction models. Empirical Software Engineering, 2012. 17(1): p. 23-33.

[67] Langdon, W.B., et al., Exact mean absolute error of baseline predictor, MARP0. Information and Software Technology, 2016. 73: p. 16-18.

[68] Zar, J.H., Biostatistical analysis. 1999: Pearson Education India.

[69] Derrac, J., et al., Analyzing convergence performance of evolutionary algorithms: A statistical approach. Information Sciences, 2014. 289: p. 41-58.

[70] Demšar, J., Statistical comparisons of classifiers over multiple data sets. The Journal of Machine learning research, 2006. 7: p. 1-30.

[71] García, S., et al., Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information sciences, 2010. 180(10): p. 2044-2064.