Int. J. Nonlinear Anal. Appl. 11 (2020) No. 2, 85-102
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2020.4384

An Intelligent Model to Predict the Development Time and Budget of Software Projects

Amid Khatibi Bardsiri*

Department of Computer Engineering, Bardsir Branch, Islamic Azad University, Bardsir, Iran.

(Communicated by Madjid Eshaghi Gordji)

Abstract

Today, software projects are a major part of information and business technology. Estimating the cost and delivery time of a project is one of the most important aspects of its development process. The results of this estimate are directly related to the failure or success of the whole project. Project anomalies, high diversity, intangibility, and poor standards are obstacles that have hampered various current forecasting models. In this regard, the aim of this paper is to provide an intelligent hybrid model that is able to weigh the project features properly. The main idea of the proposed model is based on mathematical methods, soft computing and using previous estimation methods. After the complete introduction of the new model, its efficiency is evaluated using Desharnais and ISBSG databases. The results indicate higher accuracy and greater flexibility of the proposed model. Improvement rate shows in the various metrics such as MdMRE, Pred(0.25) and MMRE separately, but the average improvement rate is 31%.

Keywords: Intelligent model, Analogy based estimation, Differential evolution algorithm, Budget and time estimation.
2010 MSC: 68T05,97R40,68T35,68N30

1. Introduction

Estimating the costs of a software project is one of the most important tasks in software projects management. The main reason for estimating such costs is that it is not possible to make accurate plans, monitor and control a project [1]. Unfortunately, the process of estimation cannot be trusted in software projects. None of the estimation models can estimate the costs of a software project

*Corresponding Author: Amid Khatibi Bardsiri
Email address: a.khatibi@srbiau.ac.ir (Amid Khatibi Bardsiri*)

Received: September 2019 *Revised:* January 2020

accurately. The estimation of human efforts in software projects is greatly influenced by irrelevant factors and misleading information. In addition, software developers are willing to improve the proposed estimates even after receiving feedback on them. One of the ways of effort estimation is to use automated estimators. In fact, an effort estimation model can be used as decision-making support tools which make it possible to investigate the effects of the features of a project and a team on the increase in costs. For the past three decades, researchers have been very willing to use various software applications for effort estimation models in order to overcome the variety of software development processes [2, 3]. Although improving the accuracy of different estimation models requires too much time and cost, effort estimation applications cannot be expected to operate highly accurately. It is because of uncertainty in software development projects and factors such as dynamic features, software innate complexities as well as the problems resulting from the lack of standardization and insufficient information on the software [4].

Software effort estimation models are meant to estimate the efforts required to develop a software application in the best way. However, the conducted effort process is not reliable because it depends on the values of features which are unclear in the first steps of the project. Nevertheless, this process is to be carried out because a huge investment is made to develop a software application. Studies show that the implementations of software projects are very unlikely to succeed, and only 30-35% of all the software projects have been completed in the determined time and budget [5]. One of the most important reasons for the failure of software projects is wrong estimation and inaccuracy. Such problems come from overestimation or insufficient effort, both of which have negative effects on the process of a software project. The use of various effort estimation models is on the rise, and software teams employ a variety of methods to estimate effort in their projects. It is important to present a realistic effort estimation without sufficient information on the area and domain of the system as well as the environmental and culture conditions of the software development team and technical complexities. The current effort estimation methods introduce the relative errors as the most important goal and try to reduce it as much as possible. Due to the uncertainty and complicated and nonlinear characteristics of software projects, the concentration of estimation model on this criterion cannot simply provide the groundwork for high and reliable accuracies. Furthermore, single-objective optimization-based estimation models are not able to manage projects. The results of such estimators are significantly different in one database from another one. Therefore, it is not possible to generalize the accuracy of the current effort estimator in various software projects because of high complexities [6, 7].

2. Related works

Estimation of software effort plays a crucial role in project development process, because it affects their project quality, turnaround time, and development risk [8]. In reality, late or early release impacts both the progress of a software product and the diversion of money. Unique computer project properties have made it tougher than it should seem [9]. Like other ventures, the software project is subjective, variable, dynamic and heterogeneous items that cannot be reliably measured, except with the assistance of a specific dataset. Different models and approaches have been suggested to approximate the effort, which can usually be divided into 2 classifications: non-algorithmic and algorithmic. For calculating the production effort, algorithmic techniques such as COCOMO (Constructive Expense MOdel), MLR (Multiple regression), ROR (Robust regression) use statistical models and variables [10]. By comparison, non-algorithmic methods like Expert Judgment (JE), RBF (Radial Basis Function) CBR (Case Based Reasoning), employ historical data analysis to estimate. Soft computation methods, expert analysis, and CBR have recently been widely used for

effort calculation in science [10, 11, 12]. For e.g., some of the items used appear in the following partial list:

- Different weighting models for software project features [11, 13, 14].
- ABE (Analogy Based Estimation) system which Shepperd implemented in 1997 [15].
- Boehm introduced the COCOMO and COCOMO II models [16, 17].
- Usage of a Bayesian network to predict software projects effort [18].
- Neural networks are used to estimate project delivery time and cost [19].
- Use of various data mining techniques to improve estimate accuracy [12].

The list is difficult to complete and there are several instances for each scenario. In 1963 expert judgement was adopted and uses the consensus of experts as a new process [20]. One effective approach is the Shepperd Analogy approach which was implemented in 1997 [21]. These models have provided the most implementations and enhancements in separate articles. Though, due to its clarity, being simple, high consistency and because it assumes no inference, ABE method has been often used and maybe it can be considered the most common estimation model [22]. Figure 1 demonstrates a number of enhancements and changes in different models to maximize the precision of the predictions.

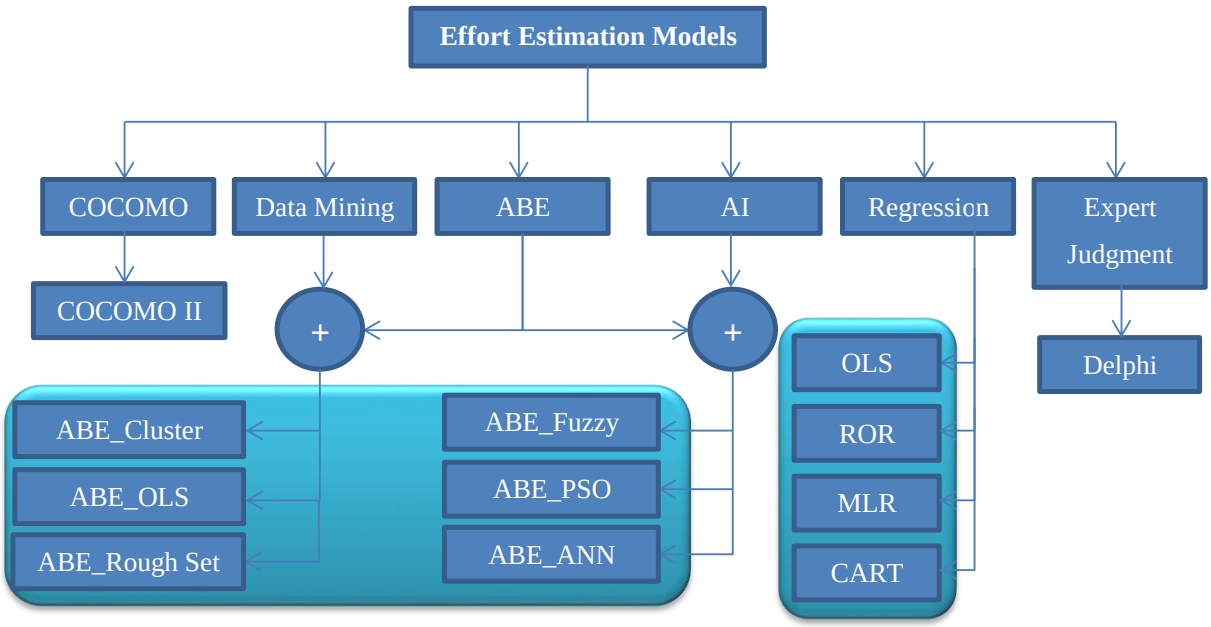

Figure 1: Various softwire development effort prediction methods

As seen in Figure 1, ABE has had the greatest enhancements as a simple and common tool and has been coupled with numerous methods such as Fuzzy theory [23], ACO (Ant Colony Optimization), and ICA (Imperialist Competitive Algorithm) [24]. While both of these approaches have resulted in some changes, but high cost, high difficulty and, most significantly, its limited implementation has diminished performance. The remainder of this paper is structured as follows: the ABE method, its essential components and the principles used in the proposed approach are presented in Part 3. Part 4 reveals Differential Evolution algorithm. Sections 5 and 6 are dedicated to developing the new model and its effects, respectively. Part 7 incorporates related development risks. Lastly, Section 8 contains future works and key findings.

3. Analogy Based Estimation

This model is generally chosen for its usability, efficiency, versatility and general researchers' experience with it. Many models or other approaches will however be used in this paper's proposed model and this is one of the benefits of the suggested approach. The ABE paradigm was implemented as an alternative to the algorithmic methods by Schofield and Shepperd in 1997 [15]. ABE identifies the connections of one project to other related projects by using the Similarity function and the final solution is sought using the Solution function after the discovery of other specific project, including the equivalences and parameters seen in KNN's parameter [11, 15, 25]. ABE is commonly used in recent studies due to its usability and reasonable approximation precision. The next steps are usually taken by this method:

- Historical data collection from prior services to create the database
- Weighting and collecting comparative mechanism functionality of a project
- Use similarity function to pick the resources closest to the test project
- Final step for the solution function to search.

3.1. Similarity Function

ABE utilizes a similarity algorithm that measures the characteristics of different digital projects and considers their correlation. There are two popular functions, Manhatan and Euclidean. Equation 3.1 indicates the Euclidean similarity function [26].

$$\begin{gathered}\operatorname{Sim}(s,s') = \frac{1}{[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i,f_{i'})}+\delta]} \\ \delta = 0.0001\end{gathered} \tag{3.1}$$

$$\operatorname{Dis}(f_i,f_i') = \left\{ \begin{array}{cc} (f_i - f_i')^2 & \text{if } f_i \text{ and } f_i' \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{array} \right\}$$

Where s and s' are the two projects contrasted and w_i is the weight of the project feature. The weight will range from 0 to 1, even f_i and f_i' show the i-th feature of s and s' projects, and n is the total features count. The worth of δ never causes the devisor to be zero. Although, other functions such as Minkowski similarity [27], maximum distance [28], and rank mean [29] are also accessible but were used less compared to the two functions provided. Equation 3.2 indicates to the Manhatan function [30].

$$\begin{gathered}\operatorname{Sim}(s,s') = \frac{1}{\left[\sum_{i=1}^{n} w_i \operatorname{Dis}\left(f_i,f_i'\right)+\delta\right]} \\ \delta = 0.0001\end{gathered} \tag{3.2}$$

$$\operatorname{Dis}(f_i,f_i') = \left\{ \begin{array}{cc} |f_i - f_i'| & \text{if } f_i \text{ and } f_i' \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{array} \right\}$$

3.2. Solution Function

The solution method is used to measure the final value of the developing project. The common solution functions are median [31], inverse weighted distance, mean and closest [24]. Median calculates the average value of project efforts with the expectation of n>2 and reverse calculate the cost value based on Equation 3.3. Mean function takes into account the average value of costs derived from n equivalent projects [32].

$$C_p = \sum_{k=1}^{K} \frac{\text{Sim}\,(s, s_k)}{\sum_{i=1}^{K} \text{sim}\,(s, s_i)} C_{S_k} \tag{3.3}$$

Where s is a new project, s_k is the related project, C_{s_k} is the s_k project effort and K is the number of most related projects in general. Eventually, $Sim(s, s_k)$ represents the degree of resemblance between the projects. For previous experiments several solution functions were used, several of which used only one solution function [15], while the others used multiple functions [33].

3.3. K Nearest Neighbor

KNN indicates the number of projects to be used in the process of comparison via ABE. Seeking the right value for K has been one of the big problems for researchers in recent years, since variations in the K value have a significant impact on the precision of the prediction. The best value for K, too, differs from one database to the next. Some experiments indicated that the value of K is fixed [31, 34] and others found it to be a dynamic value with a given range [35, 36]. Some experiments have also been published on the adaptive search approaches to determine the best values for K [32, 37, 38]. The visual structure of the ABE system is shown in Figure 2.

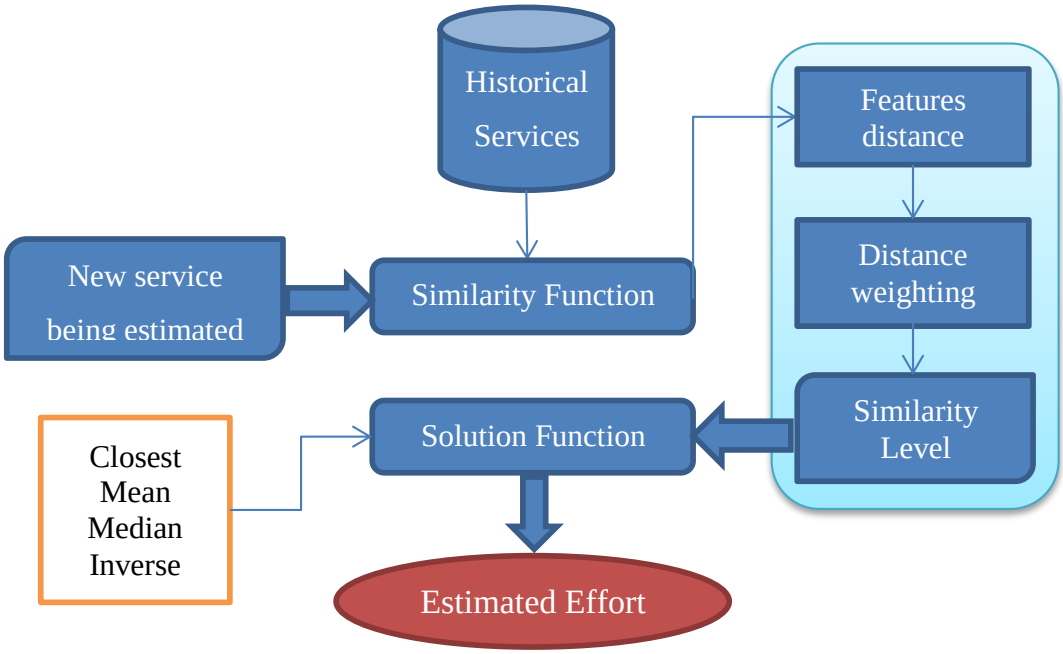

Figure 2: ABE diagram

4. Differential evolution

Differential evolution algorithm is an accurate, scalable, and simple-to-use method for mathematical optimization. DE is an optimizer based on the population that generates real encrypted vectors that depict the solutions to the problem [39]. The DE begins by an original population of real vectors. The variables are configured both arbitrarily with maybe real values, hence they are distributed uniformly across the question space. DE produces new vectors during optimizing, which are disruptions of current population vectors. The code disturbs variables with the weighted distance of two randomly population vectors and applies to a third selected vector to generate the trial vector.

The test vector interacts with the same index of the present population. If the trial produced a better option than the vector of the current population it takes its place in the dataset. Two parameters parameterise the DE algorithm. Crossover probability ($C \in [0, 1]$) defines the number of bits that are passed to the trial vector from its competitor and the scale factor ($F \in (0, 1)$) determines the frequency at which the population increases [40, 41]. Figure 3 displays the DE algorithm pseudocode. The theory of differential evolution is to get a new vector by applying the weighted difference vector of any two individuals to another [42].

1. Population initialization $X(0) \leftarrow \{x_1(0), \ldots, x_m(0)\}$
2. g ← 0
3. Compute $\{f(x_1(g)), \ldots, f(x_m(g))\}$
4. While the stopping condition is false do
5. For i = 1 to m do
6. $y_i \leftarrow$ generate Mutant$(X(g))$
7. $z_i \leftarrow$ Crossover$(x_i(g), y_i)$
8. If $f(z_i) < f(x_i(g))$ then
9. $x_i(g+1) \leftarrow z_i$
10. Else
11. $x_i(g+1) \leftarrow x_i(g)$
12. End if
13. End for
14. g ← g + 1
15. Compute $\{f(x_1(g)), \ldots, f(x_m(g))\}$
16. End while

Figure 3: A summary of differential Evolution

5. The Proposed Model

Given the variety of software projects and also the various features of software systems, comparison-based models such as ABE cannot perform well despite high simplicity and flexibility. Determining the degree of similarity that occurs between two services, irrespective of the significance of each element, can adversely affect the credibility of the analogies. The evaluation process requires more focus when compared to other projects due to the unpredictable and complicated nature of the software services. Extensive attribute evaluation may enhance the efficiency of the ABE approach before comparison with a service. As mentioned earlier, the comparison stage in the ABE approach is performed via the similarity function. Hence, the proposed method focuses optimizing similarity function efficiency. Different methods have been presented for the appropriate weighting of the available features in software projects to improve the performance of ABE. In this study, the DE algorithm was used as a tool to weight the features appropriately. Flexibility and adaptability are two valuable specifications of DE that enable it to overcome the complexity and vagueness of software project features. DE algorithm gives weight to propose similar function. The proposed model includes two steps: the training step in which appropriate weights are distributed and allocated to the features; the testing step in which the proposed model is evaluated. Since it is very complicated to compare the features of two software projects, the proposed method can improve the performance of ABE. The performance criteria introduced in this paper can be used to evaluate the proposed model.

5.1. Performance criteria

In many studies several distinct performance standards have been presented. Four Specific performance metrics are used in this study. The aim of using these parameters is to determine the accuracy of model estimations. For example, absolute residual (AR) illustrates the difference according to Equation 5.1 between real and expected values [15].

$$AR_i = \left| X_i - \widehat{X_i} \right| \tag{5.1}$$

In this equation, X_i and $\hat{X}_i$ are the actual and the predicted values of the ith instance, respectively. MRE (Magnitude of Relative Error) is an accepted, important, and extensively used performance criterion in effort estimation. As shown in Equation 5.2, MRE is the error rate between the actual required effort and the estimated one. Lower MRE values indicate better model performance [43].

$$MRE = \frac{\left| X_i - \widehat{X_i} \right|}{X_i} \tag{5.2}$$

Two other important criteria that are obtained from the overall mean and median of errors are presented in Equations 5.3 and 5.4, respectively. Here also, lower values signify better model performance. The difference between the two parameters MMRE (Mean Magnitude of Relative Error) and MdMRE (Median Magnitude of Relative Error) is that the median is less sensitive to large values and to outliers. The PRED(0.25) parameter, which is the percentage of successful predictions that fall within 25% of the actual values, is a usual substitute for MMRE and is expressed in Equation 5.5. For example, PRED(0.25) = 0.5 means that half of the estimates have a 25% distance from the actual values.

$$MMRE = \frac{\sum_{i=i}^{N} MRE}{N} \tag{5.3}$$

$$MdMRE = Median(MREs) \tag{5.4}$$

$$PRED(0.25) = \frac{100}{N} \sum_{i=1}^{N} \begin{cases} 1 & if \quad MRE_i \leq \frac{25}{100} \\ 0 & otherwise \end{cases} \tag{5.5}$$

5.2. Training Step in the Proposed Model

First, the projects were randomly divided into three classes, one of which is test, and the other two are training sets. The training projects are used to train the proposed model, whereas testing projects are used to evaluate the efficiency of the proposed model. In other words, training projects are employed to obtain the most appropriate weight and parameters. The testing projects are meant for the evaluation of the training step. The process of implementing the training step can be seen in Figure 4.

In addition, a number of weights are allocated to the features of the project with the DE algorithm. These weights are used in the similarity function through Equations 3.1 and 3.2. The number of weights will be equal to the number of features, and they are used to determine the importance of each feature. In the next step, the project (or similar projects) will be sent to the solution function. In this step, DE presents three recommendations as the solution functions, and MRE is calculated. This process is iterated until all the training projects are estimated. In other words, there will be MREs as many as the training projects. If there are no other projects, the next step will be to calculated PRED and MMRE. Therefore, the differences in the values of MMRE and PRED will be regarded as the fitness value in the proposed method.

5.3. Testing Step in the Proposed Model

Testing services assess the accuracy of the proposed method. In this step, the proposed similarity function is regarded as the inputs of the similarity function to determine the efficiency of the proposed method in training services. In addition, the optimized weights, obtained from the training step, will be sent to this function. According to Figure 5, a service is first selected from checking services and sent to the similarity function in the same way as the training step.

Then the cost required for the service is predicated. After that, MRE is calculated. This process is iterated until the testing projects are not finished. Finally, PRED and MMRE are estimated. These parameters indicate the accuracy of the proposed model. Figure 5 shows the steps of conducting the testing step.

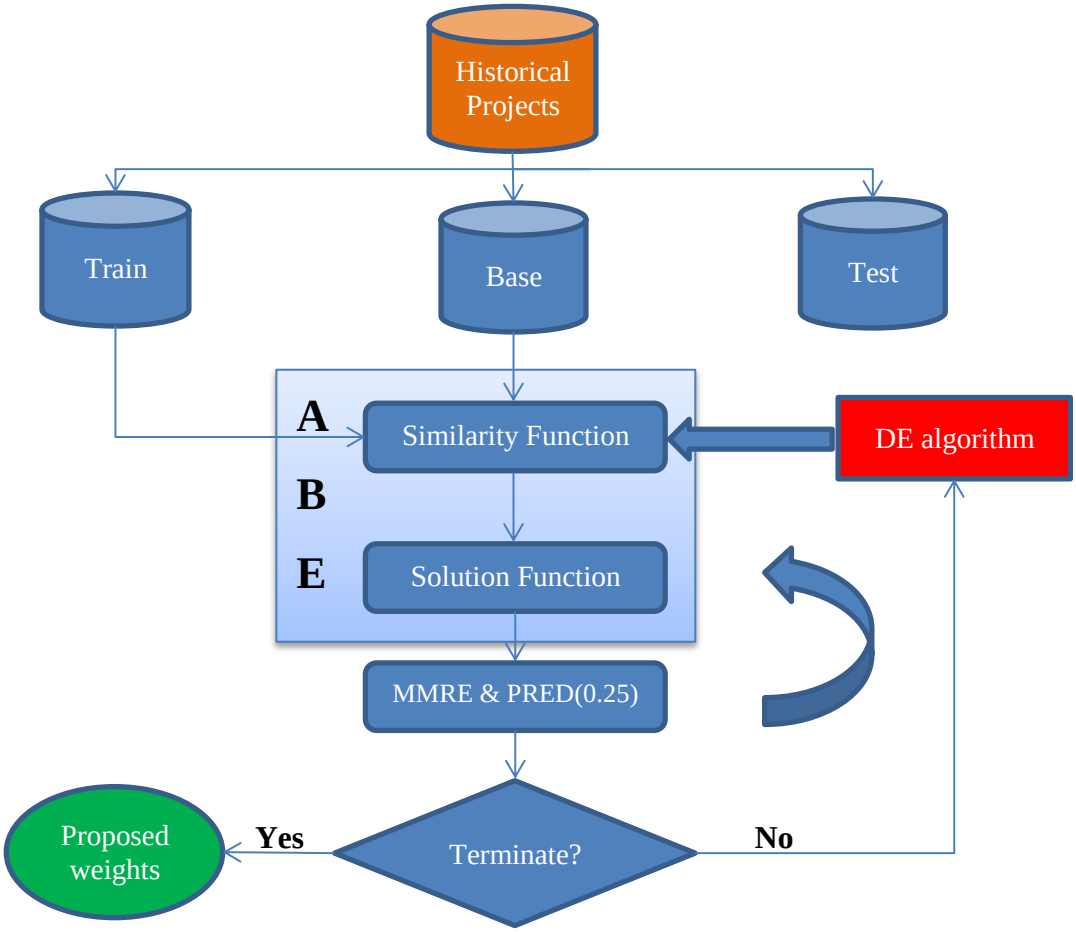

Figure 4: The flowchart of training step in the proposed model

5.4. Datasets description

The numerical data is needed for the evaluation of the prediction methods. This is why past studies used separate databases. In the preceding study, an analysis of various databases investigated the properties and reliability of each database [44]. To analyze the estimation methods, two actual databases were used. Regretfully, most existing databases are fairly small, with many anomalies included. The key benefit of artificial sets of data is they can pick and improve their size. In addition, the user can change the modes and functionality of the database.

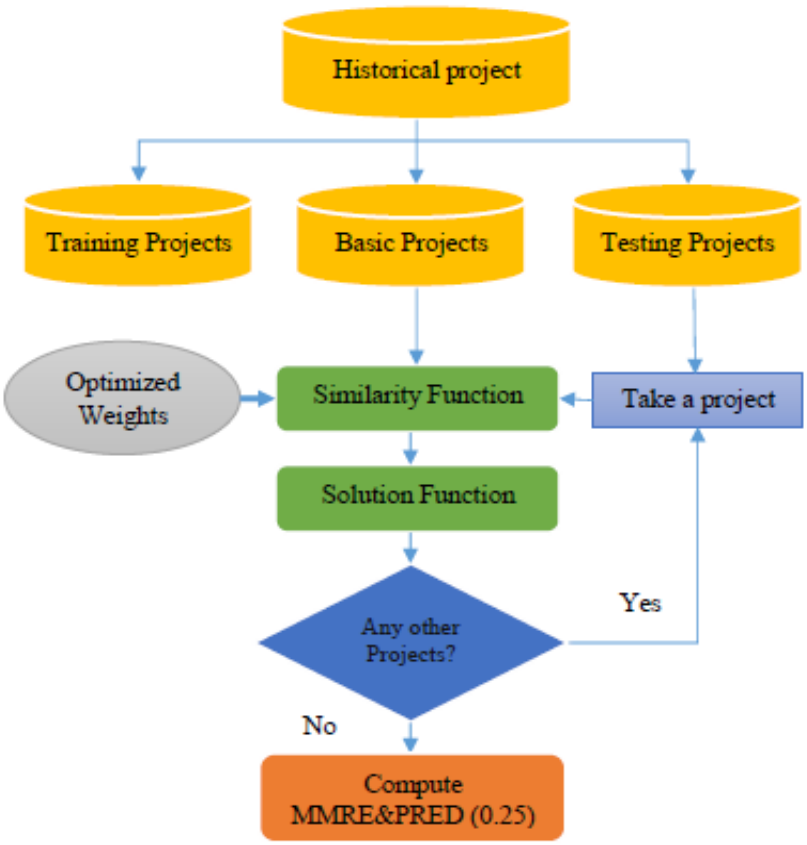

Figure 5: The flowchart of testing step in the proposed method

5.4.1. ISBSG dataset

The data in the International Software Benchmarking Standards Group (ISBSG) repository [45], which contains 5052 software projects completed in the past, was used in this research. This repository, which includes 109 features for each project, has collected its information from 24 different countries. The data contains a varied spectrum of programs, architectures, platforms, programming languages, and development tools and methods. Following the filters mentioned below, an appropriate subset of ISBSG dataset was selected for this research.

1. Among the different data in the repository, only the "a" and "b" quality rates were used that, according to the ISBSG report, there is no doubt about their correctness.
2. The normalized amount of effort was used for comparing and studying projects (in the case of incomplete projects).
3. Projects without fields that were selected for this research were omitted in order to have complete information.
4. Normalized ratios higher than 1.2 were not used (the ISBSG itself has suggested this for achieving greater accuracy).
5. A web development environment was considered that could encompass the concept of software services and the provision of web-based services.
6. The "ifpug" measurement method (the count approach) was used for all of the projects.

In the end, by following the above-mentioned filters, 66 software projects were obtained and the research was continued on them. Test data must be used to derive the efficiencies of the various

estimation models. Among all the present features, six important ones [Input count (Inpcont), Output count (Outcont), Enquiry count (EnqCont), File count (FileCont), Interface Count (IntCont) and Adjusted function point (AFP)] were selected that influenced the development effort [Normalized effort in hours (NorEffort)]. The statistical data obtained from this dataset is presented in Table 1.

Table 1: Description of ISBSG dataset

Variable	Minimum	Maximum	Mean	Median	Std
InpCont	3	1185	169	95	199
OutCont	10	698	143	67	165
EnqCont	3	653	150	116	137
FileCont	7	384	129	108	97
IntCont	5	497	76	43	95
AFP	107	2245	672	507	534
NorEffort	562	60826	6860	4899	8406

5.4.2. Dasharnais dataset

Dasharnais is one of the most common datasets in the field of software effort estimation [46]. Although this dataset is relatively old, it has been widely employed in many of recent research studies [36, 47, 48]. In this dataset, there are 81 projects related to a Canadian software company, out of which four projects include missing values and the remaining 77 projects are considered in the evaluation process. Each project is described by nine attributes. One of the attributes (language) is categorical and the remaining ones are numerical. Table 2 provides the statistical information about this dataset.

Table 2: Descriptive statistics for Desharnais dataset

Variable	Minimum	Maximum	Mean	Median	Std
TeamExp	0	4	2.30	2	1.33
ManagerExp	0	7	2.65	3	1.52
Length	1	36	11.30	10	6.79
Transactions	9	886	177.47	134	146.08
Entities	7	387	120.55	96	86.11
AdjustFactor	5	52	27.45	28	10.53
PointsAdjust	73	1127	298.01	247	182.26
Language	1	3	1.56	1	0.72
Effort (h)	546	23940	4833	3542	4188

6. Experimental results

This paper was meant to use the results of different estimation methods to compare the proposed model with other common models and obtain its accuracy. Some of these common models are ROR (robust regression), LSE (linear size adjustment), MLR (multiple linear regression), RBF (Radial basis function), SWR (stepwise regression), LMS (leas median of squares regression), ABE (analogy-based estimation) and ANN (artificial neural network). Three algorithms were separately used to evaluate the performance of the proposed model. Considering their performances, these three algorithms are very similar to each other; however, they have their own characteristics. Table 3 shows the adjustment of parameters in each algorithm.

Table 3: The initiation of parameters

Algorithms	Parameters
DE	Crossover Rate=0.1 F Constant=2 Schema=DE/Best/1
PSO	C1= C2=2 , W=1
GA	Cr=0.7 , Mr=0.1
Population size = 50, Number of iteration = 100, Fitness function =(MMRE + MdMRE)-PRED (0.25)	

6.1. ISBSG Dataset

The effort estimations of different models on ISBSG dataset indicate that this dataset can be useful for the evaluation of the proposed model. The results of the proposed model show that it did perform well on ISBSG dataset. The results were relatively good. Figure 6 indicates these observations.

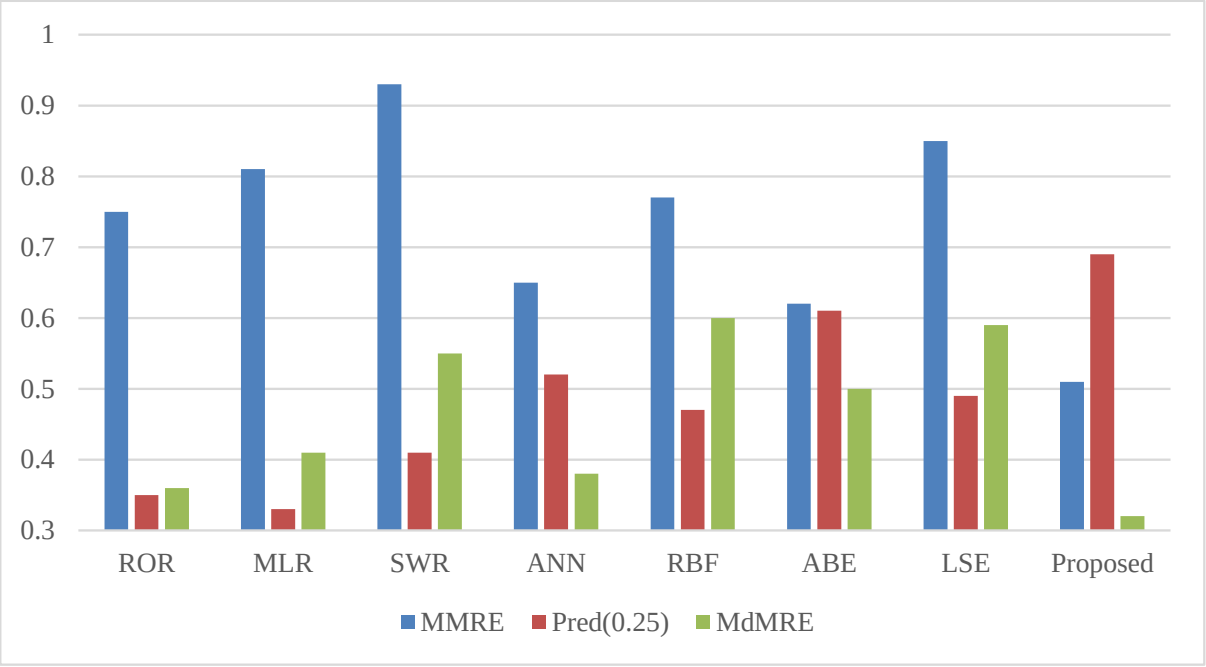

Figure 6: The results of different models on ISBSG dataset

The smallest value of MMRE was 0.51 coming from proposed model, and its worst value was 0.93 resulting from SWR. Considering this parameter in the proposed hybrid model, ANN was ranked the second, and RBF was ranked the third. Moreover, the lowest value of MdMRE was 0.32 coming from proposed method, and its worst value was 0.6 resulting from RBF. Regarding this parameter, ROR was ranked the second. Considering the last column, PRED of the best value was 0.69 coming from proposed, and the worst value was 0.34 coming from MLR. Therefore, proposed was ranked the first.

The results of the proposed model were significantly improved on this dataset mainly because of appropriate weighting and the accurate selection of solution and similarity functions. ISBSG dataset is a relatively standard dataset, a fact which has improved the accuracy of the proposed model.

As performance indicators are quantitative and generic, there is a need for an objective statistical method to provide specifics of how measurement errors are spread. Boxplot is a kind of chart which can clearly show the mean, spread, and set of values. In comparison, the Boxplot figure corresponding to MRE of different methods is seen in Figure 7. This graph illustrates how faults are spread through different prediction systems. As seen in Figure 7, the proposed method's inter-quartile interval is smaller than other ones, and from this the smallest mean is derived.

Moreover, the number of outliers in proposed method is the smallest among the models. Next to the proposed model, the SWR method yielded the best and the RBF the worst answers.

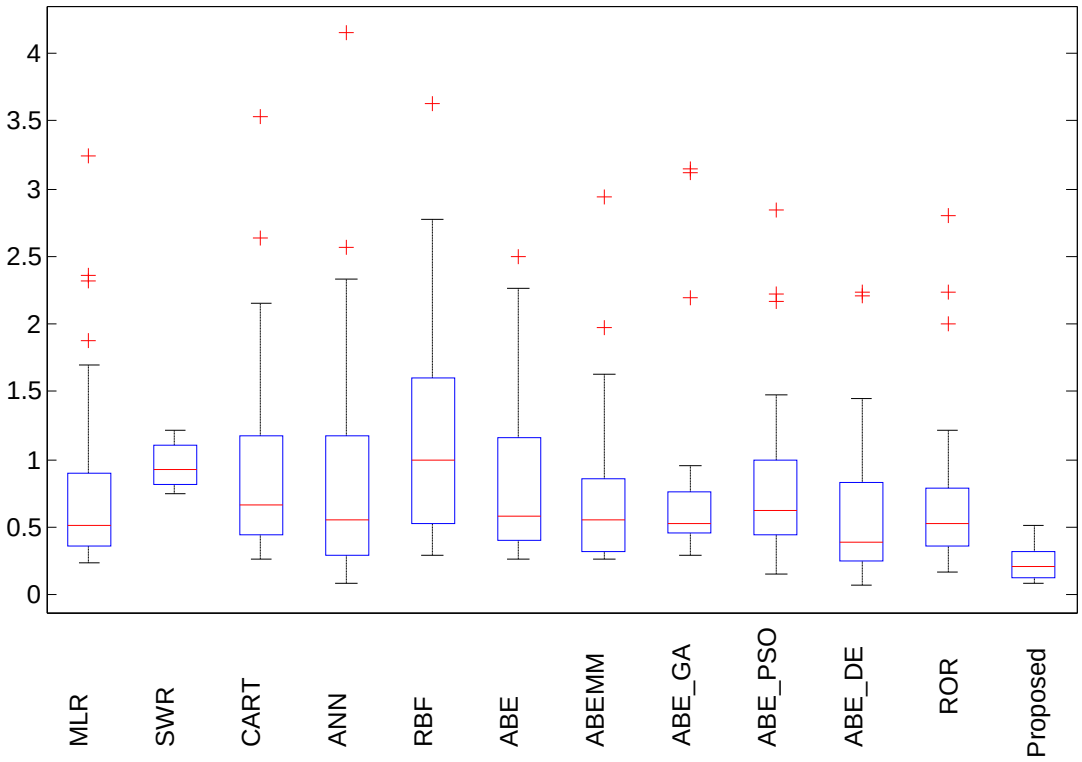

Figure 7: Dispersion of MRE in different methods in ISBSG dataset

6.2. Desharnais Dataset

This database includes 77 projects developed with the third-generation language. It also includes 8 numerical features which may influence the effort of project. The best value of MMRE was 0.37 resulting from proposed model, and the worst value was 0.64 coming from MLR. The best value of MdMRE was 0.31 coming from proposed method. ABE was ranked the second with 0.39; however, its worst value was 0.55 coming from LSE. The best value of PRED was 0.81 resulting from proposed model. ANN was ranked the second with 0.72, and its worst value was 0.32 coming from ROR. The results indicated that the proposed model performed very well on Desharnais dataset, and its accuracy was appropriate. Figure 8 indicates the results.

As shown in Figure 9, the proposed model values had the least MdMRE, BMMRE, and MMRE relative to the other approaches and, moreover, provided the maximum PRED(0.25). The two Grey and SWR approaches did the poorest. Given this dataset's low heterogeneity and its structure, all the measurements reported here are fairly acceptable, and there's very little variation between the methods. The Boxplot diagram in Figure 9 illustrates how the MRE values for the different methods of cost estimation is dispersed. Here also, the least median and inter-quartile belonged

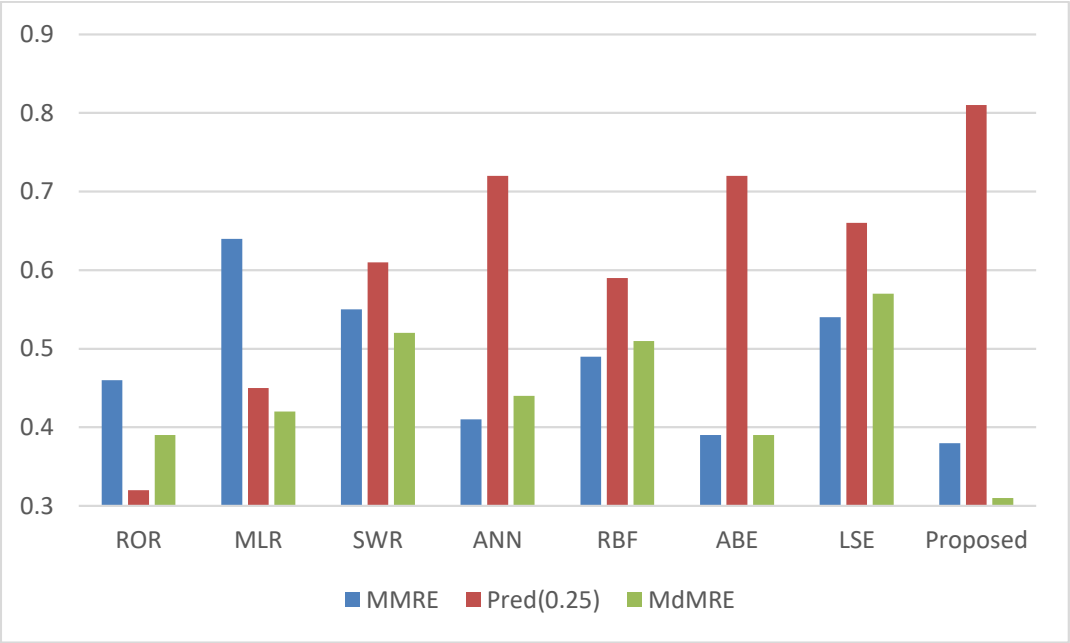

Figure 8: The results of different model on Desharnais dataset

to the proposed model and, next to it, MLR and ANN had suitable error distribution and small medians. Considering the larger number of projects in this dataset, a greater number of outliers can be seen in the Boxplot diagram. This figure shows how various models estimate and indicates their efficiencies well.

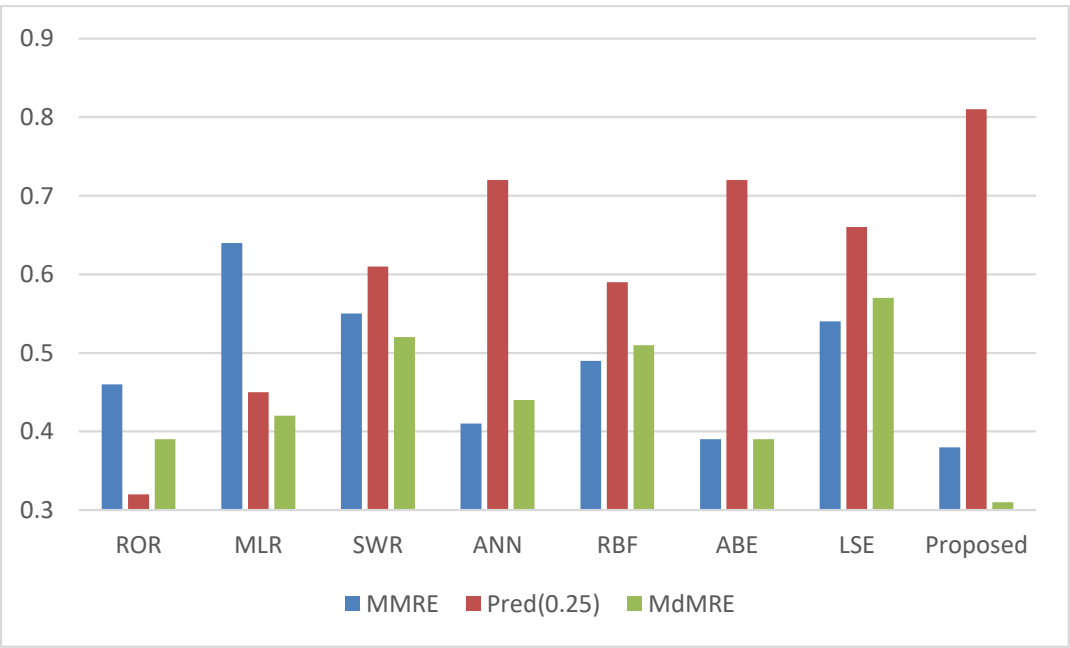

Figure 9: Dispersion of MRE in different methods in Desharnais dataset

7. Threats to the validity

Any scientific study or analytical research will encounter certain external and internal challenges. In reality, for their outcomes to be accurate and functional, methods must be properly understood and exploited. Any of the external and internal risks are discussed in this part. The problem of its correct and precise measurement is one of the key challenges to the credibility of an effort assessment method. For example, it is a really dangerous idea to use a portion of the training data to check the model as the output that is achieved would not be true. Another bad policy is to find a specific number of projects for evaluating the system, when the best thing to do is to use approaches such as fold cross or leaving one out. In this method each service has both the learning roles and the test; and, for this reason, these methodological methods have been used very often in previous studies.

Another challenge is the application of evaluation criteria: they must be consciously considered without selective filtering. In this paper we used four of the most common metrics, since estimating accuracy was our intention. Since these parameters are commonly used, their use helps us to conveniently compare the outcomes produced with those of other experiments and, in fact, helps us to use mathematical models such as the Box diagram to help us see the uncertainty and distribution of the responses. The next element is the model construction process, including how variables can be modified and weighted in dynamic systems. For example, what kind of solution and similarity works, and what importance to use for KNN. Configuration and modification of variables are defined in detail in this article. There would be numerous changes to different structures, and thus different responses. The most noticeable potential challenges in future experiments will be the way the theoretical method is implemented and applied in the actual world. For the proposed method, all drawbacks and threats should be considered so that adequate outcomes can be achieved. For eg, the proposed method is not based on any single database, and will fully adjust to specific data and project styles. In comparison, it has no clear approach for sorting and filtering attributes and essential instances in a database.

8. Discussion

This study introduced an intelligent method for improving of degree the precision of budget prediction. ABE approach had been paired with the differential evolution algorithm in the proposed system. In the first step, the DE determines suitable weights to be used in the similarity of the ABE model, and then those numbers can be used in the testing phase to analyze new services. DE proposed the most appropriate similarity and solution functions as well as the most optimized weights and selected the number of the closest neighbors in ABE to improve the effort estimation accuracy. Authors compared the findings with the best prediction models of the few years particulary RBF and mixed CBR. Since then, deferential evolution has been used to adjust the discrepancy between a current service and its analogies with regard to all service properties. The basic concept of using DE is to maximize the weight factor of each element gap using one fitness function and LSE method in an effort to change closest analogies based on scale analysis, it shows strong results compared to ABE however this approach was validated over very small databases.

Proposed model showed a better performance than existing methods Moreover, the mean magnitude of relative error and percentage of error prediction were used to determine the accuracy of the proposed model. Two datasets were used to evaluate the results of the proposed model. The proposed model could produce appropriate results in any of the three performance parameters. The results of the proposed model were significantly improved on Desharnais dataset mainly because of appropriate weighting and the accurate selection of solution and similarity functions. Desharnais dataset is a relatively standard dataset, a fact which has improved the accuracy of the proposed

model. The results indicate that the proposed model performed very well on this Dataset, and it was appropriately accurate. This model's popularity is for the close connection between effort and size, and the availability of a suitable size property. The proposed method has a high versatility and durability, as it needs no preconditions and expectations to build and use it.

The proposed model can be useful in improving the performance of effort estimation methods. According to our research, key findings are as below list:

1. ABE method has not good performance if it uses alone. So, it should be combined with an evolutionary algorithm such as PSO, GA or DE.
2. Authors study and simulate eleven different method to estimate software project effort. The proposed method can perform better than others.
3. Authors use two different datasets to achieve good validation. This case doesn't see in the previous researches.
4. The authors combine ABE method with evolutionary algorithm as a new idea. Results achieved indicate that the latest approach introduced is better than all previous studies. For instance, suppose decision trees, regression methods, analogy-based methods and soft computing algorithms.
5. Improvement rate shows in the various figures separately. But the average rate is 31%. This value indicates more precision and correctness in estimation process.
6. Finally, our comparison is the strongest review for other researchers. They can use these promise results to improve their works and companies.

9. Conclusion

The successful management of a project plays an effective role in the productivity of an organization. Effort estimation, required to create an information system, is one of the major concerns of project management. Analogy based estimation method is one of the most successful techniques to software effort estimating project. ABE is used as the main benchmark here for the reason of simplicity of implementation, are clear and clarity of functioning for the user. ABE method alone has a low precision that this defect can be overcome by creating a hybrid model.

In the ABE approach, the efficiency of the comparative procedure was increased by applying the most fitting weights to service attributes. Two databases were used to assess the quality of the proposed method, and MdMRE, Pred(0.25) and MMRE performance indicators were calculated using a cross-validation methodology. The results obtained were compared with popular prediction methods which revealed the supremacy in two databases of the proposed method. It can be inferred that the synthesis of ABE and differential evolution algorithm leads to a high-performance approach in terms of estimating the software development cost, based on the findings obtained from two datasets. This method is a reliable, scalable, and efficient prediction framework, ideal for use in different software services styles. Combine the Analogy based estimation method and metaheuristics such as GA and DE algorithm to achieve new model and suitable performance respect to the obtained results, the performance of the proposed method is acceptable.

References

[1] Prokopova, Z., et al. (2019). Analysis of the Software Project Estimation Process: A Case Study. Computer Science On-line Conference, Springer.
[2] Sehra, S. K., et al. (2017). "Research patterns and trends in software effort estimation." Information and Software Technology 91: 1-21.

[3] Sharma, P. and J. Singh (2017). Systematic literature review on software effort estimation using machine learning approaches. 2017 International Conference on Next Generation Computing and Information Systems
[4] Bardsiri, A. K., Seyyed Mohsen Hashemi, and Mohammadreza Razzazi (2016). "GVSEE: A Global Village Service Effort Estimator to Estimate Software Services Development Effort." Applied Artificial Intelligence 30(5): 396-428.(ICNGCIS), IEEE.
[5] Menzies, T., et al. (2017). "Negative results for software effort estimation." Empirical Software Engineering 22(5): 2658-2683.
[6] Ceke, D. and B. Milasinovic (2015). "Early effort estimation in web application development." Journal of Systems and Software 103: 219-237.
[7] Huijgens, H., et al. (2017). Effort and Cost in Software Engineering: A Comparison of Two Industrial Data Sets. Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, ACM.
[8] Goyal, S. and P. K. Bhatia (2019). A Non-Linear Technique for Effective Software Effort Estimation using Multi-Layer Perceptrons. 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), IEEE.
[9] Sangwan, O. P. (2017). Software effort estimation using machine learning techniques. 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE.
[10] Sigweni, B. and M. Shepperd (2014). Feature weighting techniques for CBR in software effort estimation studies: a review and empirical evaluation. Proceedings of the 10th International Conference on Predictive Models in Software Engineering, ACM.
[11] Wu, D., et al. (2013). "Linear combination of multiple case-based reasoning with optimized weight for software effort estimation." The Journal of Supercomputing 64(3): 898-918.
[12] Benala, T. R., et al. (2014). Software Effort Estimation Using Data Mining Techniques. ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I, Springer.
[13] Wen, J., et al. (2009). Improve analogy-based software effort estimation using principal components analysis and correlation weighting. Proceeding of the 16th Asia-Pacific Software Engineering Conference, Penang, Malaysia, IEEE.
[14] Li, Y.-F., et al. (2009). "A study of project selection and feature weighting for analogy based software cost estimation." Journal of Systems and Software 82(2): 241-252.
[15] Shepperd, M. and C. Schofield (1997). "Estimating software project effort using analogies." IEEE Transactions on Software Engineering, 23(11): 736-743.
[16] Boehm, B., et al. (2000). "Software development cost estimation approaches-A survey." Annals of software engineering 10(4): 177-205.
[17] Boehm, B. W. (1981). Software engineering economics, Prentice-hall Englewood Cliffs (NJ).
[18] Aggarwal, K., et al. (2005). "Bayesian regularization in a neural network model to estimate lines of code using function points." Journal of Computer Sciences 1(4): 505-509.
[19] Nassif, A. B., Mohammad Azzeh, Luiz Fernando Capretz, and Danny Ho (2016). "Neural network models for software development effort estimation: a comparative study." Neural Computing and Applications 27(8): 2369-2381.
[20] Jorgensen, M. (2005). "Practical guidelines for expert-judgment-based software effort estimation." IEEE Software, 22(3): 57-63.
[21] Shepperd, M. and C. Schofield (1997). "Estimating Software Project Effort Using Analogies." IEEE Transactions on Software Engineering, 23(11): 736-743.
[22] Bardsiri, A. K. and S. M. Hashemi (2014). "Software Effort Estimation: A Survey of Well-known Approaches." International Journal of Computer Science Engineering (IJCSE) 3(1): 46-50.

[23] Shivakumar, N., N. Balaji, and K. Ananthakumar (2016). "A Neuro Fuzzy Algorithm to Compute Software Effort Estimation." Global Journal of Computer Science and Technology 16(1).
[24] Keung, J. W. and B. Kitchenham (2007). Optimising project feature weights for analogy-based software cost estimation using the mantel correlation. 14th Asia-Pacific Software Engineering Conference, APSEC 2007. , IEEE.
[25] Song, Q. and M. Shepperd (2011). "Predicting software project effort: A grey relational analysis based method." Expert Systems with Applications 38(6): 7302-7316.
[26] Amazal, F.-A., et al. (2014). An Analogy-Based Approach to Estimation of Software Development Effort Using Categorical Data. Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA), Krakow, Czech Republic, IEEE.
[27] Idri, A., et al. (2015). "Analogy-based software development effort estimation: A systematic mapping and review." Information and Software Technology 58(Article in press): 206-230.
[28] Malathi, S. and S. Sridhar (2012b). "Performance evaluation of software effort estimation using fuzzy analogy based on Complexity." International Journal of Computer Applications 40(3): 32-37.
[29] Kocaguneli, E., et al. (2012a). "Exploiting the essential assumptions of analogy-based effort estimation." IEEE Transactions on Software Engineering, 38(2): 425-438.
[30] Azzeh, M., et al. (2011). "Analogy-based software effort estimation using Fuzzy numbers." Journal of Systems and Software 84(2): 270-284.
[31] Li, J. and G. Ruhe (2006). A comparative study of attribute weighting heuristics for effort estimation by analogy. Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, Rio de Janeiro, Brazil, ACM.
[32] Leung, H. K. (2002). "Estimating maintenance effort by analogy." Empirical Software Engineering 7(2): 157-175.
[33] Chiu, N.-H. and S.-J. Huang (2007). "The adjusted analogy-based software effort estimation based on similarity distances." Journal of Systems and Software 80(4): 628-640.
[34] Huang, S.-J. and N.-H. Chiu (2006). "Optimization of analogy weights by genetic algorithm for software effort estimation." Information and Software Technology 48(11): 1034-1045.
[35] Auer, M. and S. Biffl (2004). Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. Proceedings of the International Symposium on Empirical Software Engineering, IEEE Computer Society, Washington DC, USA.
[36] Firouziana, Iman, Morteza Zahedia, and Hamid Hassanpoura. "Real-time Prediction and Synchronization of Business Process Instances using Data and Control Perspective." Int. J. Nonlinear Anal. Appl 10, no. 1 (2019): 217-228.
[37] Walkerden, F. and R. Jeffery (1999). "An empirical study of analogy-based software effort estimation." Empirical Software Engineering 4(2): 135-158.
[38] Angelis, L. and I. Stamelos (2000). "A simulation tool for efficient analogy based cost estimation." Empirical Software Engineering 5(1): 35-68.
[39] Khatibi Bardsiri, A., and Seyyed Mohsen Hashemi (2016). "A differential evolution-based model to estimate the software services development effort." Evolution and Process 28(1): 57-77.
[40] Hu, Z., et al. (2014). "A convergent differential evolution algorithm with hidden adaptation selection for engineering optimization." Mathematical Problems in Engineering 2014: 1–18.
[41] Mohanty, B., et al. (2014). "Controller parameters tuning of differential evolution algorithm and its application to load frequency control of multi-source power system." International Journal of Electrical Power & Energy Systems 54: 77-85.
[42] Storn, R. and K. Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces." Journal of global optimization 11(4): 341-359.

[43] Shepperd, M. and G. Kadoda (2001). "Comparing Software Prediction Techniques Using Simulation." IEEE Transactions on Software Engineering, 27(11): 1014-1022.
[44] Mair, C., et al. (2005). An analysis of data sets used to train and validate cost prediction systems. ACM SIGSOFT Software Engineering Notes, ACM.
[45] ISBSG (2011). International Software Benchmarking standard Group.
[46] Desharnais, J.-M. (1989). "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction." University of Montreal.
[47] Li, Y.-F., et al. (2009b). "A study of project selection and feature weighting for analogy based software cost estimation." Journal of Systems and Software 82(2): 241-252.
[48] Jodpimai, P., et al. (2010). Estimating software effort with minimum features using neural functional approximation. Computational Science and Its Applications (ICCSA), 2010 International Conference on, IEEE.