Cluster Computing (2019) 22:S11329–S11338
https://doi.org/10.1007/s10586-017-1388-0

An effective software project effort estimation system using optimal firefly algorithm

V. Resmi[1] · S. Vijayalakshmi[2] · R. Subash Chandrabose[3]

Received: 20 August 2017 / Revised: 23 October 2017 / Accepted: 15 November 2017 / Published online: 6 December 2017

Abstract
The software effort estimation is one of the active presentations in the software project administration. Accordingly, it is not frequently possible to antedate the exact guesses in the estimation of software development effort. There are many techniques used for effort estimation. But we cannot confirm that one particular method alone gives good accuracy in estimates. In this expose, a hybrid process is gracefully boosted for the estimation of the effort of software project. The innovative process is unknown; but consolidation of the fuzzy analogy by the side of the firefly and the Expectation-Maximization (EM) process that is envisaged for estimation of the software project lead to the enhancement of accuracy in prediction. Furthermore, an EM is employed to group large amount of data. The significant production is set as an input to the fuzzy analogy in parallel to the Firefly Algorithm (FA). Consecutively, the FA is competently familiar in enhancing the optimal solutions and thereby improves estimation accuracy. The fuzzy analogy reliably helps the presentation of assessing the effort of the software project. The epoch-making process is proficient in java platform and its task is competently estimated.

Keywords Expectation maximization · Fuzzy analogy · Firefly algorithm · Software effort estimation

1 Introduction

The goal of software engineering is to develop the techniques and tools needed to develop high-quality applications that are more stable and maintainable. In order to assess and improve the quality of an application during the development process, developers and managers use several metrics [1]. For various business and technical motives such as shorter development cycles, lower development costs, improved product quality, and access to source code more and more software developers and companies are basing their software products on open source components [2]. Estimating software development cost remains a complex problem, and one which continues to attract considerable research attention. Improving the accuracy of the cost estimation models available to project managers would facilitate more effective control of time and budgets during software development. The need for reliable and accurate cost estimation in software engineering was an ongoing challenge for software engineers in the last decade. In order to make accurate estimates and avoid large errors, several cost estimation techniques have been proposed [3].

The ability to accurately and consistently estimate software development efforts, especially in the early stages of the development life cycle, is required by the project managers in planning and conducting software development activities because the software price determination, resource allocation, schedule arrangement and process monitoring are dependent upon it. This issue lies in the fact that software development is a complex process due to the number of factors involved, including the human factor, the complexity of the product that is developed, the variety of development platforms and the difficulty of managing large projects [4]. For effective project management such as budgeting, project planning and control, accurate software development cost estimation is important. Until now, no model has proved to be completely successful at effectively and consistently predicting software development cost. To estimate software development effort the use of the neural networks has been viewed with skepticism by the best part of the cost estima-

✉ V. Resmi
resmi.nandakumar@gmail.com

[1] Department of Computer Applications, Sun College of Engineering and Technology, Erachakulam, India

[2] Department of Computer Applications, Thiagarajar college of Engineering, Madurai, India

[3] Sun College of Engineering and Technology, Nagercoil, India

tion community. Although, neural networks have shown their strengths in solving complex problems, their limitation of being 'black boxes' has forbidden them to be accepted as a common practice for cost estimation [5].

Software cost estimation techniques can be broadly classified as algorithmic and non-algorithmic models. Algorithmic models are derived from the statistical analysis of historical project data, for example, constructive cost model (COCOMO) and software life cycle management (SLIM). Non-algorithmic techniques include Price-to-Win, Parkinson, expert judgment and machine learning approaches. Machine learning is used to group together a set of techniques that embody some of the facets of human mind, for example fuzzy systems, analogy, regression trees, rule induction neural networks and evolutionary algorithms. Among the machine learning approaches, fuzzy systems and neural networks and Evolutionary algorithms are considered to belong to the soft computing group [6].

During the development process, the cost and time estimates are useful for the initial rough validation and monitoring of the project's completion process. And in addition, these estimates may be useful for project productivity assessment phases [7]. The limitations of algorithmic models led to the exploration of the non algorithmic techniques which are soft computing based. These include

- Artificial neural network
- Evolutionary computation
- Fuzzy logic models
- Case-based reasoning and
- Combinational models [8].

Accurate cost estimation is important because of the following reasons,

- It can help to classify and prioritize development projects with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support re-planning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs [9].

Recently, software effort estimation methods have been grouped into three categories such as algorithmic models (AM), expert judgment (EJ), and machine learning (ML) process [10,11]. But as designated by Boehm, software effort estimation approaches are prepared into six categories: parametric representation, decision tables, learning focused measures which integrate the case-based thinking (CBR) approach, regression based procedure; progression based process and composite procedure [12]. Nowadays, on the principle of these procedures, abundant software cost estimation tools have been produced. The enormous preponderance of these automated tools rely on the dimension measures, for example, lines of code (LOC) and function point (FP).

This document is prearranged as follows: a concise assessment of some of the literature works in the software cost evaluation is presented in Sect. 2. The incentive for this study is specified in Sect. 3. Section 4 elucidates the concise explanation for the anticipated methodology. The investigational outcome and presentation study deliberations are offered in Sect. 5. At last, the conclusion is in Sect. 6.

2 Related work

Plentiful studies have been conducted by researchers for the evaluation of Software cost. We have also studied the fundamentals of software costing and pricing. Divergent processes of cost evaluation should be exploited if project manager evaluates costs. Following are only some literatures to employ for evaluation of the state-of-art work on the evaluation of software cost. A number of researches have been proposed by researchers for the estimation of Software cost. We have also analyzed the fundamentals of software costing and pricing. Different techniques of cost estimation should be used when estimating costs. Following are a few literatures applied for assessment of the state-of-art work on the estimation of software cost.

Early stage software effort estimation was a crucial task for project development and feasibility studies. Since collected data during the early stages of a software development lifecycle was always imprecise and uncertain, it was very hard to deliver accurate estimates. Analogy-based estimation, which was one of the popular estimation methods, was rarely used during the early stage of a project because of uncertainty associated with attribute measurement and data availability. Azzeh et al. [13] have integrated analogy-based estimation with Fuzzy numbers in order to improve the performance of software project effort estimation during the early stages of a software development lifecycle, using all available early data. Particularly, a software project similarity measure and adaptation technique based on Fuzzy number was proposed. Empirical evaluations with Jack-knifing procedure have been carried out using five benchmark data sets of software projects, namely, ISBSG, Desharnais, Kemerer, Albrecht and COCOMO, and results are reported. The results were compared to those obtained by methods employed in the literature using case-based reasoning and stepwise regression. In all data sets the empirical evaluations have shown that the proposed similarity measure and adaptation techniques method were able to significantly improve the per-

formance of analogy-based estimation during the early stages of software development. The results have also shown that the proposed method outperforms some well known estimation techniques such as case-based reasoning and stepwise regression.

Izzat Alsmadi and Hassan Najadat [14] have proposed an approach towards the ability to predict software fault modules and the ability to correlate relations between faulty modules and product attributes using statistics. Correlations and relations between the attributes and the categorical variable or the class are studied through generating a pool of records from each dataset and then selecting two samples every time from the dataset and comparing them. The correlation between the two selected records was studied in terms of changing from faulty to non-faulty or the opposite for the module defect attribute and the value change between the two records in each evaluated attribute (e.g. equal, larger or smaller). The goal was to study if there are certain attributes that are consistently affecting the change of the state of the module from faulty to none, or the opposite. Results indicated that such technique could be very useful in studying the correlations between each attribute and the defect status attribute. Another prediction algorithm was developed based on statistics of the module and the overall dataset. The algorithm gave each attribute true class and faulty class predictions. They found that dividing prediction capability for each attribute into those two (i.e. correct and faulty module prediction) facilitates understanding the impact of attribute values on the class and hence improve the overall prediction relative to previous studies and data mining algorithms. Results were evaluated and compared with other algorithms and previous studies. ROC metrics were used to evaluate the performance of the developed metrics.

Estimating the work-effort and the schedule required to develop and/or maintain a software system was one of the most critical activities in managing software projects. Software cost estimation was a challenging and onerous task. Estimation by analogy was one of the convenient techniques in software effort estimation field. However, the methodology used for the estimation of software effort by analogy was not able to handle the categorical data in an explicit and accurate manner. Different techniques have, so far, been used like regression analysis, mathematical derivations, simulation, neural network, genetic algorithm, soft computing, fuzzy logic modelling etc. Ziauddin et al. [15] have aimed at utilizing soft computing techniques to improve the accuracy of software effort estimation. In this approach fuzzy logic was used with particle swarm optimization to estimate software development effort. The model has been calibrated on 30 projects, taken from NASA dataset. The results of this model are compared with COCOMO II and Alaa Sheta Model. The proposed model yields better results in terms of MMRE.

Khatibi Bardsiri et al. [16] have proposed a hybrid method to increase the accuracy of development effort estimation based on the combination of fuzzy clustering, ABE and ANN methods. In the proposed method, the effect of irrelevant and inconsistent projects on estimates was decreased by designing a framework, in which all the projects were clustered. Two relatively large datasets were employed to evaluate the performance of the proposed method and the obtained results were compared to eight other estimation methods. These methods were selected from the most common algorithmic and non-algorithmic methods used extensively in the field of software development effort estimation. All comparisons were performed based on MMRE and PRED (0.25) parameters using three-fold cross validation technique. According to the obtained results, the proposed method outperformed the other methods and significantly improved the accuracy of estimates in both datasets.

The effort invested in a software project was probably one of the most important and most analyzed variables in recent years in the process of project management. The limitation of algorithmic effort prediction models was their inability to cope with uncertainties and imprecision surrounding software projects at the early development stage. More recently attention has turned to a variety of machine learning methods, and soft computing in particular to predict software development effort. Soft computing was a consortium of methodologies centering in fuzzy logic, artificial neural networks, and evolutionary computation. It was important, to mention here, that these methodologies are complementary and synergistic, rather than competitive. They provide in one form or another flexible information processing capability for handling real life ambiguous situations. These methodologies are currently used for reliable and accurate estimate of software development effort, which has always been a challenge for both the software industry and academia. Sehra et al. [17] was to analyze soft computing techniques in the existing models and to provide in-depth review of software and project estimation techniques existing in industry and literature based on the different test datasets along with their strength and weaknesses.

Software development effort estimation was a daunting task that was being carried out by software developers, as much of the information about the software was not available during the early stages of development. The information that was to be gathered for various attributes of software needs to be subjective which otherwise leads to imprecision and uncertainty. Inaccurate estimation of the software effort and schedule leads to financial losses and also delays in project deadline. Sandeep Kad and Vinay Chopra [18] have presented the use of soft computing technique to build a suitable model which improves the process of effort estimation. To do so, various parameters of constructive cost model (COCOMO) II are fuzzified that leads

to reliable and accurate estimates of effort. The results showed that the value of Magnitude of Relative Error (MRE) obtained by applying fuzzy logic was quite lower than MRE obtained from algorithmic model. By analyzing the results further it was observed that Gaussian Membership Function (gaussmf) performs better than Triangular Membership Function (trimf) and Trapezoidal Membership Function (trapmf) as the transition from one interval to another was quite smoother.

Software estimation accuracy was one of the greatest challenges for software developers. Formal effort estimation models, like COCOMO are limited by their inability to manage uncertainties and impression surrounding software projects early in the project development cycle. A software effort estimation model which adopts a soft computing technique provides a solution to adjust the uncertain and vague properties of software effort drivers. Brajesh Kumar Singh and Misra [19] have proposed a model in which COCOMO was used as algorithmic model and an attempt was being made to validate the soundness of artificial neural network technique, using NASA project data, in order to investigate the effect of crisp inputs and soft computing technique on the accuracy of system's output. Proposed model was validated by using 85 NASA project dataset. Empirical results showed that application of the ANN model for software effort estimates resulted in slightly smaller mean magnitude of relative error (MMRE) and probability of a project having a relative error of less than or equal to 0.25 as compared with results obtained with COCOMO was improved by approximately 17.54%.

The main difficulty in the software effort estimation process is the selection of estimation models. We cannot use all the models for that. And also these models do not handle missing and noisy data. These models are purely based on mathematical formula. So if any data is missing, estimation process is becoming a challenging one. Benala, Dehuri, Mall, suggests some of the computational intelligence models, such as artificial neural networks and fuzzy systems for effort estimation to meet the above problem [20]. Benala, Mall, Dehuri and Prasanthi, use the combined fuzzy C-Means clustering algorithm and functional link artificial neural networks (FLANN) to achieve accurate software effort prediction [21].

Ali Idri et al. [22] suggests two approaches : classical analogy and fuzzy analogy. His research reports that fuzzy analogy produces better results than classical analogy. Ali Idri et al. [23] inspects the treatment of missing data through two analogy-oriented software effort evaluation processes: classical analogy and fuzzy analogy. The conclusion projected that fuzzy analogy produced more accurate assessment than classical analogy even though some data are missing.

3 Problem definition

Software effort estimation is a system to predict the majority legitimate quantity of effort requirement to develop software on the source of imperfect, indefinite and insufficient input data. Sequentially, to carry out cost-benefit research, cost assessment is to be attained by client or developer. The mutual concern in existing software evaluation procedure is specified beneath,

- Software cost assessment is a multifarious activity that necessitates knowledge of a number of key qualities that affect the consequences of software projects. The most serious issue is the lot of information that needed is frequently incredible to get in desirable quantities.
- One among the main issues is effort estimation accuracy.
- Enormous effort estimation approaches have been projected, the accuracy of estimation is not sustaining and the attempts continue to progress the function of assessment approaches.
- The foremost purpose for the project catastrophe of the software effort estimation is inaccuracy of the estimation model.
- Understanding and selection of models for effort estimation on the basis of historical information are difficult because of inherent multifaceted relationships within the associated attributes.

These are the leading shortcomings of plenty of obtainable installation that encourages us to carry out the research and to work with the proposed approach.

Objectives

- The primary intention of this research is to estimate the effort accurately
- To estimate the effort, cluster the input attribute values to differentiate the High and low effort projects.
- The optimal rules will be used for accurate effort estimation
- To minimize the MRE with the help of fuzzy analogy with optimization process.
- To estimate the effort efficiently, MMRE is used as a fitness for optimization process.

4 Proposed method

In the software project organization, the software effort estimation has been modernized as one of the lively presentations. By means of a characteristic of a project to estimate the effort of the software, an innovative process on the fuzzy analogy connected through the firefly algorithm is proficiently

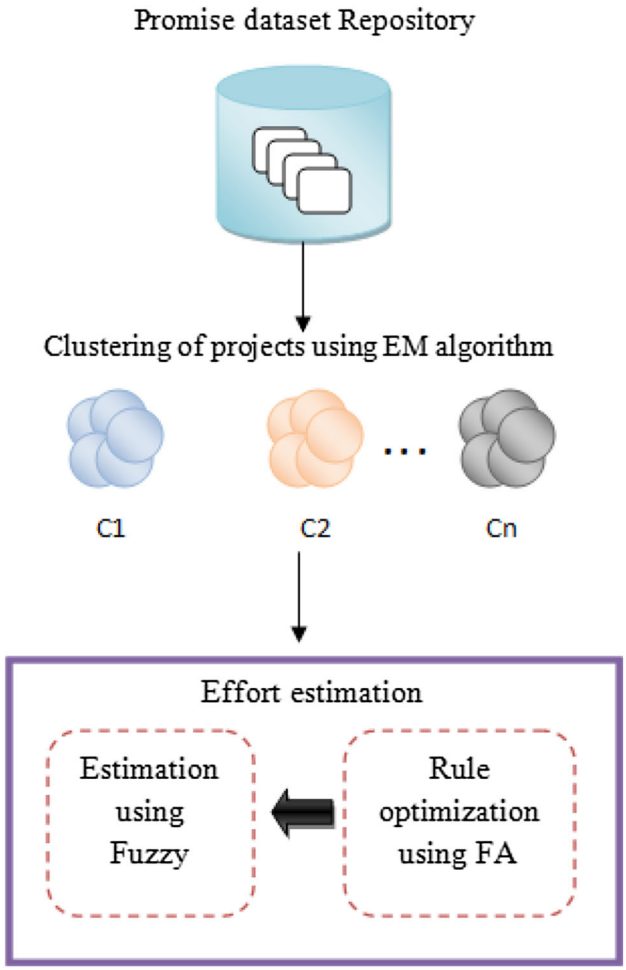


Fig. 1 Block diagram for proposed methodology

derived. Principally, we decide on the input dataset from the promise data repository. In our projected process, we have chosen four datasets such as nasa 93, nasa 60, cocomo81 and deshnaris from the repository. The modus operandi(a method of procedure) of our objective procedure is vividly portrayed in the block diagram presented in Fig. 1 beneath.

4.1 Expectation maximization (EM) algorithm for clustering

It is unsupervised method, which does not need any training phase; it tries to find the parameter of the probability distribution that has the maximum likelihood of its parameters. Its main role is to parameter estimation. It is an iterative method, which is mainly used to finding the maximum likelihood parameters of the model. The E-step involves the computation of cluster membership probabilities. The probabilities calculated form E-Step is estimated with the parameters in M-Step. Here the EM algorithm is used to cluster the data obtained from the input data.

Assuming that the parameters of each component are represented by a parameter vector E_m, the problem is to determine the values of the components of this vector, and this can be achieved using the expectation maximization algorithm. Following random initialization of the parameter vectors E_m, $m = 1, 2, \ldots, C$ an Expectation step (E-step) followed by a maximization step (M-step) are iterated until convergence. The E-step computes the cluster membership probabilities. For example, assuming spherical Gaussian mixture components, these probabilities are calculated using equation as follows.

$$P(m/Xi) = \frac{\pi_m \mathrm{P}\,(X_i/\beta_m, \sigma_m)}{\sum_{k=1,\ldots C} \pi_k \mathrm{P}\,(X_i/\beta_m, \sigma_m)} \tag{1}$$

β_m, σ_m are the existing evaluation of the mean and standard deviation, likewise, of element m. the denominator proceed as a normalization factor, authenticate the value as follows

$$0 \leq P(m|X_i) \leq 1 \text{ and } \sum_{m=1}^{C} P(m|X_i) = 1$$

In the M-step, these probabilities are then used to re-estimate the parameters. The spherical Gaussian case, the following equation are used to evaluate the likelihood clusters.

$$\beta_m = \frac{\sum_{i=1}^{N} P(m|X_i) X_i}{\sum_{i=1}^{N} P(m|X_i)}, \tag{2}$$

$$\sigma_m^2 = \frac{\sum_{i=1}^{N} P(m|X_i) \, \|X_i - \mu_m\|^2}{\sum_{i=1}^{N} P(m|X_i)} \tag{3}$$

$$\pi_m = \sum_{i=1}^{N} P(m|X_i) \tag{4}$$

At this time, we attained the clustered data that will be further augmented for better estimation of effort. In our work, the data are clustered into three clusters using the above EM clustering algorithm.

4.2 Effort estimation using fuzzy analogy with firefly algorithm

The Fuzzy logic is an unconventional procedure, well-geared to accomplish at suitable consequences to the difficulties encumbered by means of complexities that cannot be comprehended quantitatively. A fuzzy set symbolizes a relationship presentation that compared with every spot in the fuzzy set a genuine integer in the period (0, 1), marked a level or score of relationship. The relationship presentation can be distinguished into three varieties like the triangular, trapezoidal and Gaussian.

The roadmap of the Fuzzy logic comprises three stages as illustrated below:

1. Fuzzification
2. Fuzzy rule-based system
3. Defuzzification

Stage 1: Fuzzification: In this step, a crisp input is improved into a fuzzy set
Stage 2: Fuzzy rule-based system: In this scheme, fuzzy IF-THEN rules are employed.

Fuzzy inference engine: After all the crisp input values are fuzzified into their related linguistic values, the inference engine gets into the fuzzy rule base to arrive at the linguistic values for the intermediate and the output linguistic variables.

Stage 3: Defuzzification: In this stage, the fuzzy output is transformed into the crisp output.

4.2.1 Fuzzy analogy

The fuzzy analogy [24] characterizes a 'fuzzification' of the traditional comparison process. It flows through three phases as follows: identification of cases, retrieval of similar cases and case adaptation.

Step 1: Identification of a case

The primary motivation after this phase conveys to the association of complete software projects by a cluster of uniqueness. Each software project is precise by a cluster of elected characteristics that are resolute by linguistic values. For that, let us carry out that there are M attributes and for each attribute (M_i) normalize the linguistic variable (L_v). Each linguistic variable is measured by a fuzzy set through the relationship function (M_f). The fuzzy set and their relationship presentations are elected by computerized and practical techniques. In the feature of computerized technique the relationship presentation is structured from the historical information. On an additional feature, practical technique accomplishes the relationship presentation by the use of professional facts. In our innovative technique we pursue the invention of the relationship presentation from the historical information.

4.2.2 Rule optimization by firefly algorithm

Firefly algorithm is a bio-inspired metaheuristic algorithm for optimization problems. It was introduced in 2009 at Cambridge University by Yang [25,26]. The algorithm is inspired by the flashing behavior of fireflies at night. The three rules behind this algorithm are

- All fireflies are unisex, which means any firefly can be attracted to any other brighter one.
- Brightness of a firefly is determined from the objective function.
- Attractiveness is directly proportional to brightness but decreases with distance, and a firefly will move towards the brighter one, and if there is no brighter one it will move randomly.

In the original firefly algorithm we estimate the fitness value of each firefly. The fitness value of each i th firefly is compared with the j th adjacent firefly. When the fitness value of adjacent firefly is better than the current one, we evaluate the distance between every firefly in the population by means of Euclidean distance measure. At present, this distance is employed to evaluate the attractiveness (A).

$$A = A_0 e^{-\gamma d_{ij}^2} \tag{5}$$

where,

A_0 —denotes to the preset attractiveness
γ —characterizes the light absorption coefficient
d_{ij} —represents the distance between ith firfly and jth neighboring firefly.

If a firefly located at p_{jx} is brighter than p_{ix} then the firefly p_{ix} will move towards p_{jx} (6). The movement of p_{ix} is done by the following Eq. (6).

$$p_{ix} = p_{ix} + A(p_{jx} - p_{ix}) + \alpha(\delta - 1/2) \tag{6}$$

The last term is a randomization term where α is a a randomization parameter with values $0 \leq \alpha \leq 1$ and δ is vector of random numbers. The second term is for the attraction of p_{ix} towards p_{jx}.

The complete process is repeated till the stopping criteria is met.

In the modern process, we systematize the firefly algorithm to adjust the formed morality. At this occasion, each outcome is erratically arrived at surrounded by explicit search space. Furthermore, each outcome is labeled as regulations P_{ix}, where $P_{ix} = \{p_{i1}, p_{i2}, \ldots p_{iy}\}$. Mainly, the fitness value of each regulation is evaluated. The regulations that meets the outstanding fitness values are certified as the existing finest regulations. For each firefly (rule), fitness value is appraised and the summation of MMRE of each regulation is considered as the finest fitness value. Based on these fitness values, regulations are ranked. Finally the attractiveness is the minimum MMRE value.

$$Attractiveness = \min \sum_{x=1}^{y} MMRE \tag{7}$$

In our work, regulation with the higher MMRE is replaced by regulation with least MMRE.

The whole process is gracefully established in the flowchart Fig. 2.

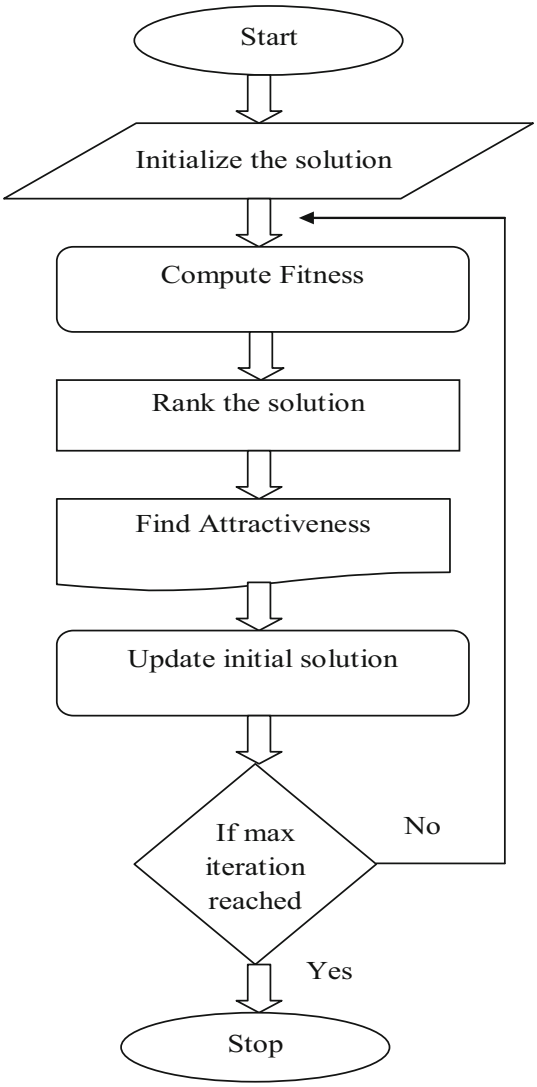


Fig. 2 The flowchart for firefly algorithm

Step 2: Retrieval of similar cases

This step focuses on the process of finding similar projects. This is noteworthy as it will manipulate similar analogies. In our anticipated procedure, candidate measures are used to select the set of analogies. These measures review the common resemblance of two projects P_1 and P_2, $d\,(P_1, P_2)$ by integration the entire discrete resemblance of P_1 and P_2 associated to the frequent linguistic variables V_j labeling P_1 and $P_2, d_{V_j}\,(P_1, P_2)$. Subsequently an axiomatic justification of a few expected candidate dealings for the individual measures $d_{V_j}\,(P_1, P_2)$, we include occupied two measures.

$$d_{V_j}\,(P_1, P_2) = \begin{cases} \max\limits_{\mathrm{k}} \quad \min(\mu_{A_k^j}(P_1), \mu_{A_k^j}(P_2)) \\ \max - \min \quad aggregation \\ \sum\limits_{k} \mu_{A_k^j}(P_1) \times \mu_{A_k^j}(P_2) \\ sum - product \quad aggregation \end{cases} \tag{8}$$

where V_j are the linguistic variable relating the project P_1 and P_2. A_k^j are the fuzzy sets related to V_j and $\mu_{A_k^j}$ are the membership functions signifying fuzzy sets A_k^j. The ambiguity of the cost drivers reminiscently anguishes the exactness of the endeavor guesstimate resultant from software effort evaluation representation. As the indistinctness and ambiguity of software effort drivers cannot be circumvented, a fuzzy representation encompasses the enhancement of easily verifying the cost drivers by presumptuous fuzzy sets.

$$Effort = A*(SIZE)^{B+0.01*\sum_{i=1}^{N} d_i} * \prod EM_i \tag{9}$$

where A and B are constants, $\mathrm{d_i}$ is distance measure and EM is effort multipliers. The cost drivers are fuzzified by triangular and trapezoidal fuzzy sets for every linguistic value like very low, low, nominal, high etc. as suitable to every cost driver. Regulations are mechanized by cost driver in the predecessor segment and conforming endeavor multiplier in the significant segment. The defuzzified value of each of the endeavor multiplier is obtained from individual fuzzy inference systems subsequent to identical, inference aggregation and resultant Defuzzification. Complete EAF is accomplishing subsequent to multiplying them collectively. The elevated values for the cost drivers escort an endeavor review that is more than three times the introduction guesstimate(an estimate based on a mixture of guesswork and calculation.), while low values weaken the evaluation to about one third of the inventive.

Step 3: Case adaptation

The elemental cause at the back of this segment is to improve a guesstimate for the original project by the system of the predictable endeavor values of indistinguishable projects. At this instance, we are dealing with tackle of two fundamental difficulties. Each of the historical project sponsors its tick in the evaluation of the endeavor of the original project, on the origin of the stage of similarity through the related project.

4.3 Effort estimation

The paramount purpose of this function is generally to evaluate the dimensions of the software product. In this regard, there are two leading kinds of cost evaluation techniques such as the algorithmic and non-algorithmic methods. The algorithmic techniques, in turn, are capable of incredible modification in the statistical elegance. Certain methods are dependent on easy arithmetic formulas employing summary statistics like the mean and standard deviations. Some other approaches invariably rely on the regression models together with the differential equations. The primary measure to evaluate the cost is to arrive at the expenditure needed for the purpose of procurements. The subsequent stage constitutes

the evaluation of the cost of the training intended for the software project. Hence, the procedure for evaluating the effort and cost of software product may be carried out by means of the EM algorithm as well as the Fuzzy analogy with FA technique. The comprehensive test outcomes assessment is represented by Sect. 5 appearing below.

5 Results and discussion

The anticipated procedure is implemented in JAVA. The dataset exploited in the revision is NASA 93, NASA 60, COCOMO 81 and deshnaris datasets.

5.1 Data set description

The dataset exploited in the revision is NASA 60. The NASA 60 dataset includes 60 whole projects, enclosing 17 autonomous variables of which 15 are unconditional.

The NASA 93 dataset includes 93 whole projects, enclose 17 autonomous variables of that 15 are unconditional.

COCOMO 81 dataset comprises of 81 projects which are having 17 attributes and 63 instances.

Desharnais dataset, which contains 81 projects among incompleteness in 4 projects was detached. This data set prepared openly accessible in order to support repeatable, confirmable, refutable, and/or improvable analytical representation of software engineering. There are nine autonomous variables and one dependant variable in the dataset.

5.2 Performance analysis

To measure the accuracy of the estimated effort generated by the proposed method we use the following metrics. General procedure for reviewing the endeavor evaluation is mean magnitude of relative error (MMRE), and prediction (PRED).

MMRE

The MMRE can be measure by the subsequent formula,

$$MMRE = \frac{1}{n}\sum_{i}^{n} MRE_i \tag{10}$$

$$MRE = \frac{|act_{effort} - est_{effort}|}{|act_{effort}|} \tag{11}$$

PRED

$$PRED\,(25) = \frac{k}{n} \tag{12}$$

k —The number of observations whose MRE is less or equal to 25

n —Number of observation

Table 1 MMRE and prediction (25) calculation

Dataset	MMRE	Pred (.25) %
Deshnaris	0.04101	79.63
Cocomo81	0.156	81
Cocomonasa60	0.0781	85.5
Cocomonasa93	0.0562	88.25

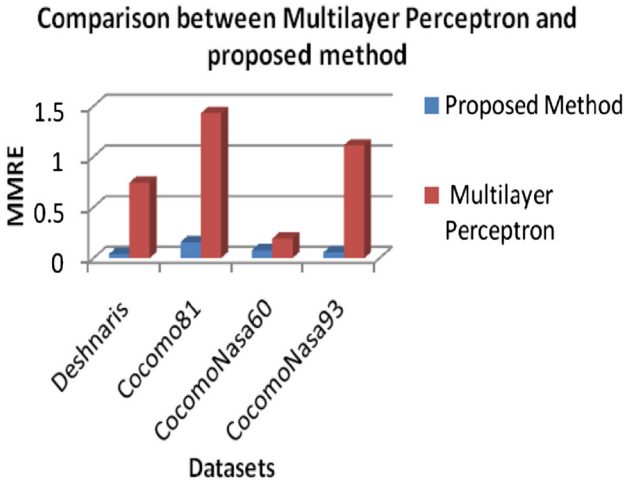


Fig. 3 Comparison between multilayer perceptron and proposed method based on MMRE

The MMRE and Prediction (25) metrics for all the four datasets are given in Table 1.

5.3 Comparative analysis of proposed work with some of the available existing techniques

The comparative analysis of our proposed technique is compared with one of the existing techniques, multilayer perceptron.

The following chart Fig. 3 shows the comparison between multilayer perceptron and proposed method based on MMRE.

From the above chart Fig. 3, it is clear that MMRE values for all the four datasets based on our proposed method are much less than the MMRE values for the same datasets based on multilayer perceptron.

The following chart Fig. 4 shows the comparison between Multilayer Perceptron and proposed method based on Pred(25).

From the above chart Fig. 4, it is clear that prediction values for all the four datasets based on our proposed method are much less than the prediction values for the same datasets based on multilayer perceptron.

The comparative analysis of our proposed method is compared with various existing techniques.

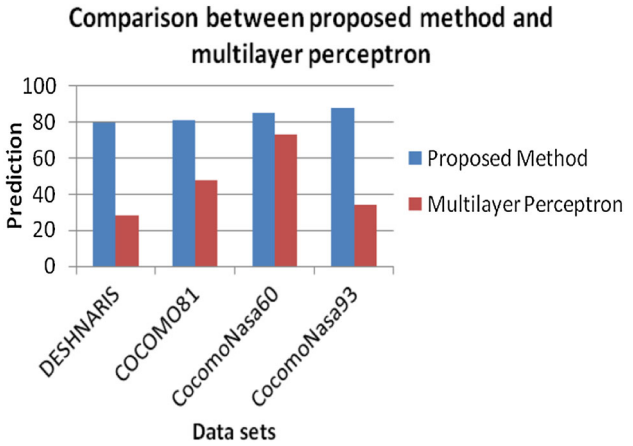


Fig. 4 Comparison between multilayer perceptron and proposed method based on prediction

Table 2 Comparison of MMRE with multilayer perceptron

Datasets	Techniques	
	Proposed method	Multilayer perceptron
Deshnaris	0.04101	0.72
Cocomo81	0.156	1.43
Cocomo nasa60	0.0781	0.19
Cocomo nasa93	0.0562	1.11

Table 3 Comparison of pred (25) with multilayer perceptron

Datasets	Techniques	
	Proposed method	Multilayer perceptron
Deshnaris	79.63	28.3
Cocomo81	81	47.6
Cocomo nasa60	85.5	73
Cocomo nasa93	88.25	34

Table 4 MMRE comparison for deshnaris dataset

Techniques	MMRE
Proposed method	0.04101
Multilayer perceptron	0.72
Analogy based estimation [27]	34.3
Fuzzy analogy [28]	0.0498

Table 5 MMRE comparison for Cocomo81 dataset

Techniques	MMRE
Proposed method	0.156
Multilayer perceptron	1.43
Analogy based estimation [27]	29.3

Table 6 MMRE comparison for Cocomonasa60 dataset

Techniques	MMRE
Proposed method	0.0781
Multilayer perceptron	0.19
Fuzzy analogy [28]	0.05

Table 7 MMRE comparison for Cocomonasa93 dataset

Techniques	MMRE
Proposed method	0.0562
Multilayer perceptron	1.11
Soft computing techniques[29]	0.026
Fuzzy analogy [28]	0.069

From Tables 2, 3, 4, 5, 6 and 7, it is evident that our proposed method for effort estimation with the fusion of clustering, optimization and fuzzy analogy produces good results.

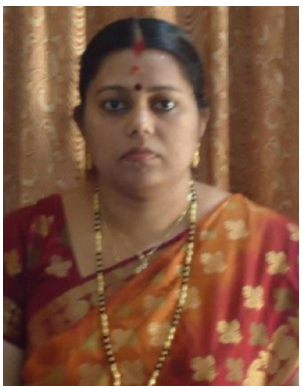
6 Conclusion

In this critique, the modern fusion procedure based fuzzy with firefly and EM algorithm is gracefully launched for predicting the software project effort. At present, the fusion of fuzzy and firefly algorithm on clustered datasets is exploited to improve the accuracy of the estimation. The Firefly algorithm gives optimal set of rules and this will help in improving the accuracy in the estimation process. So, we are delighted to make a note that our fascinating process comes out through flying colors in understanding superb consequences concerning the parallel process. It is anticipated that the future researchers will be able to execute their own implementations through their individual outstanding optimization process so as to improve the effort estimation accuracy.

References

1. Al Dallal, J.: Mathematical validation of object-oriented class cohesion metrics. Int. J. Comput. **4**(2), 45–52 (2010)
2. Orsila, H., Geldenhuys, J., Ruokonen, A., Hammouda, I.: Update propagation practices in highly reusable open source components. In: Proceedings of 20th World Computer Congress on Open Source Software, Milano, Italy, vol. 275, pp. 159–170 (2008)
3. Attarzadeh, I., Ow, S.H.: A novel soft computing model to increase the accuracy of software development cost estimation. In: Proceedings of 2nd International Conference on Computer and Automation Engineering (ICCAE), vol. 3, (2010)
4. Attarzadeh, I., Ow, S.H.: Proposing a new software cost estimation model based on artificial neural networks. In: Proceedings of 2nd International Conference on Computer Engineering and Technology, vol. 3, pp. 487–491 (2010)

5. Idri, A., Khoshgoftaar, T.M., Abran, A.: Can neural netwoks be easily interpreted in software cost estimation?", 2002 World Congress on computational intelligence, Honolulu, Huwaii, pp. 1–8, May 12–17, (2002)
6. Hari, C.H.V.M.K., Jagadeesh, P.R. Ganesh,G.S:Interval type-2 fuzzy logic for software cost estimation using TSFC with mean and standard deviation. In: Proceedings of International Conference on Advances in Recent Technologies in Communication and Computing, (2010)
7. Yadav, R.K., Niranjan, S.: Software effort estimation using fuzzy logic: a review. Int. J. Eng. Res. Technol. (IJERT) **2**(5), 1377–1384 (2013)
8. Merugu, R.R.R., Dammu, V.R.K.: Effort estimation of software project. Int. J. Adv. Res. Comput. Eng. Technol. **1**(10), 34–41 (2012)
9. Zia, Z., Rashid, A., uz Zaman, K.: Software cost estimation for component based fourth-generation-language software applications. IET Softw. **5**(1), 103–110 (2011)
10. Kaushik, A., Soni, A.K., Soni, R.: An improved functional link artificial neural networks with intuitionistic fuzzy clustering for software cost estimation. Int. J. Syst. Assur. Eng. Manag **7**(1), 1–12 (2014)
11. Batra, G., Barua, K.: A review on cost and effort estimation approach for software development. Int. J. Eng. Innov. Technol. **3**(4), 290–293 (2013)
12. Bardsiri, V.K., Jawawi, D.N.A., Hashim, S.Z.M., Khatibi, E.: A PSO-based model to increase the accuracy of software development effort estimation. Softw. Qual. J **21**(3), 501–526 (2013)
13. Azzeh, M., Neagu, D., Cowling, P.I.: Analogy-based software effort estimation using fuzzy numbers. J. Syst. Softw. **84**, 270–284 (2011)
14. Alsmadi, I., Najadat, H.: Evaluating the change of software fault behavior with dataset attributes based on categorical correlation. Adv. Eng. Softw. **42**, 535–546 (2011)
15. Ziauddin, S.K.T., Zaman, K., Zia, S.: Software cost estimation using soft computing techniques. Adv. Inf. Technol. Manag **2**(1), 233–238 (2012)
16. Khatibi Bardsiri, V., Jawawi, D.N.A., Hashim, S.Z.M., Khatibi, E.: Increasing the accuracy of software development effort estimation using projects clustering. IEEE Trans. IET Softw. **6**(6), 461–473 (2012)
17. Brar, Y.S., Kaur, N.: Soft computing techniques for software project effort estimation. Int. J. Adv. Comput. Math. Sci. **2**(3), 160–167 (2011)
18. Kad, S., Chopra, V.: Software development effort estimation using soft computing. Int. J. Mach. Learn. Comput. **2**(5), 548 (2012)
19. Singh, B.K., Misra, A.K.: An alternate soft computing approach for efforts estimation by enhancing constructive cost model in evaluation method. Int. J. Innov. Manag. Technol. **3**(3), 272 (2012)
20. Benala, T.R., Dehuri, S., Mall, R.: Computational intelligence in software cost estimation: an emergingparadigm. ACM SIGSOFT Softw. Eng. Notes **37**(3), 1–7 (2012)
21. Benala, T.R., Mall, R., Dehuri, S., Prasanthi, V.L.: Software effort prediction using fuzzy clustering and functional link artificial neural networks. In International Conference on Swarm, Evolutionary, and Memetic Computing (pp. 124-132). Springer, Berlin (2012)
22. Idri, A., Hosni, M., Abran, A.: Improved estimation of software development effort using classical and fuzzy analogy ensembles. Appl. Soft. Comput. **49**, 1–55 (2016)
23. Idri, A., Abnane, I., Abran, A.: Missing data techniques in analogy-based software development effort estimation. J. Syst. Softw. **117**, 1–23 (2016)
24. Malathi, S., Sridhar, S.: Optimization Of fuzzy analogy in software cost estimation using linguistic variables. International Conference on Modeling, Optimization and Computing (ICMOC-2012), (2011)
25. Yang, X.S.: Nature-Inspired Metaheuristic Algorithm, 2nd edn. Luniver Press, Beckington, UK (2010)
26. Tilahun, S.L., Ong, H.C.: Modified firefly algorithm. J. Appl. Math., Hindawi Publishing Corporation, Vol. 2012, Article ID 467631, pp. 12, https://doi.org/10.1155/2012/46763
27. Azzeh, M. Nassif, A. B.: Analogy-based effort estimation: a new method to discover set of analogies from dataset characteristics. IET Software, https://doi.org/10.1049/iet-sen.2013.0165
28. Malathi, S., Sridhar, S.: Estimation of effort in software cost analysis for heterogenous dataset using fuzzy analogy. Int. J. Comput. Sci. Inf. Secur. **10**(10), (2012)
29. Shanker, M., Jaya, J., Thanushkodi, K.: An effective approach to software cost estimation based on soft computing techniques. Int. Arab. J. Inf. Technol. **12**(6), 1–12 (2015)



V. Resmi is working as Assistant Professor in the department of Computer Applications and research scholar of Anna University, Chennai. Published few papers in the international level conferences. Interested in doing research in the area of data mining and software project management.

S. Vijayalakshmi is working as Assistant Professor in the department of Computer Applications in Thiagarajar College of Engineering. Madurai. Published several research papers in various reputed journals. She has supervised several Ph.D scholars from different universities. Has conducted many workshops for scholars and students. She is a certified IBMWebsphere application developer and external member of board of governers in Taminadu polytechnic college, Madurai. Published several books.

R. Subash Chandrabose is working as a professor and principal of a reputed engineering college of Tamilnadu. Published several research papers in various journals. Has guided research scholars from different universities.