Asian Journal of Information Technology 15 (9): 1484-1493, 2016
ISSN: 1682-3915

Analogy Based Software Effort Estimation Based on Differential Evolution and Hybrid Fuzzy Logic and Firefly Algorithm

[1]I. Thamarai and [2]S. Murugavalli
[1]Department of Computer Science, Panimalar Polytechnic, Chennai, India
[2]Department of Computer Science, Panimalar Engineering College, Chennai, India

Abstract: The software effort evaluation has surfaced as one of the vital functions in the software project management and therefore it is always not feasible to anticipate the precise estimates in the upgrade of software. In this regard, the vital constraints to be taken into account for the software effort evaluation encompass the size of the project, schedule and number of person concerned. In this document, a hybrid technique is elegantly launched for the evaluation of the effort of the software product. The innovative approach is nothing but the amalgamation of the fuzzy analogy with the firefly and the Differential Evolution (DE) technique which is intended for evaluation of the effort and considerable decrease in the error rate. Further a differential evolution approach is employed to create the rules for the performed technique. The consequential output is furnished as input to the fuzzy analogy with the firefly algorithm. The Firefly Algorithm (FA), in turn is effectively utilized to optimize the rules and scale down the error rate. The Fuzzy analogy faithfully discharges the function of evaluating the effort of the software. The epoch-making technique is performed in java platform and its performance is effectively assessed.

Key words: Software effort estimation, fuzzy analogy, differential evolution, firefly algorithm, epoch-making technique

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 (Al Dallal, 2010). 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 (Orsila *et al*., 2008). 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 (Attarzadeh and Ow, 2010).

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 (Attarzadeh and Ow, 2010). 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 unbeaten 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 estimation 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 (Idri *et al*., 2002).

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

Corresponding author: I. Thamarai, Department of Computer Science, Panimalar Polytechnic, Chennai, India

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 (Hari, *et al*., 2010).

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 (Yadav and Niranjan, 2013).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 (Merugu *et al*., 2012)

Accurate cost estimation is important because of the following reasons:

- It can help to classify and priorities 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 (Zia and Rashid, 2011)

Literature review: 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 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 bedding 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*. (2011) 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 performance of analogy-based estimation during the early stages of software development. The results have also shown that the proposed method outperforms some well know estimation techniques such as case-based reasoning and stepwise regression.

Alsmadi and Najadat (2011) 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 select two samples every time from the dataset and compare 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 changing 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) facilitate 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*. (2012) have aimed to utilize 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.

The proposed a hybrid method to increase the accuracy of development effort estimation based on the combination of fuzzy clustering, ABE and ANN methods (Khatibi *et al*., 2012). 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*. (2011) 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 not much of the information about the software which was 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 loses and also delays in project deadline. Kad and Chopra (2012) 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 development estimation accuracy was one of the greatest challenges for software developers. Formal effort estimation models, like Constructive Cost Model (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. Singh and Misra (2012) 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 when proposed model applied to the NASA dataset derive the software effort estimates. Proposed model 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 <0.25 as compared with results obtained with COCOMO was improved by approximately 17.54%.

Problem definition: Software effort estimation is the process of predicting the most realistic amount of effort required to develop or maintain software based on incomplete, uncertain and noisy input. In order to perform cost-benefit analysis, cost estimation is to be performed by client or developer. The common problem in existing software estimation method is given below:

- Software cost estimation is a complex activity that requires knowledge of a number of key attributes that affect the outcomes of software projects. The most critical problem is the lot of data is needed, which is often impossible to get in needed quantities
- One of the major challenges is effort estimation accuracy
- Numerous effort estimation methods have been proposed, the accuracy of estimates is not satisfying and the attempts continue to improve the performance of estimation methods
- The main reason for the project failure of the software effort estimation is imprecision of the estimation model
- Understanding and calculation of models based on historical data are difficult due to inherent complex relationships between the related attributes

These are the main drawbacks of various existing work which motivate us to do the research on analogy based software effort estimation.

MATERIALS AND METHODS

The software effort estimation has emerged as one of the vital functions in the software project management and hence it is always not feasible to anticipate precise evaluations in the development of the software. With a view to ascertain d the effort estimation of the software a novel technique dependent on the fuzzy analogy coupled with directional evolution algorithm and the firefly algorithm is proficiently launched. Initially, we choose the input dataset from the database In our eye-catching technique, we have followed the NASA dataset as the input dataset which is initiated on the differential evolution technique for configuring the rules which are subsequently furnished as the input to fuzzy analogy couple with the firefly algorithm for evaluating the effort of the software product. The modus operandi of our milestone method is colorfully pictured in the block diagram shown in Fig. 1.

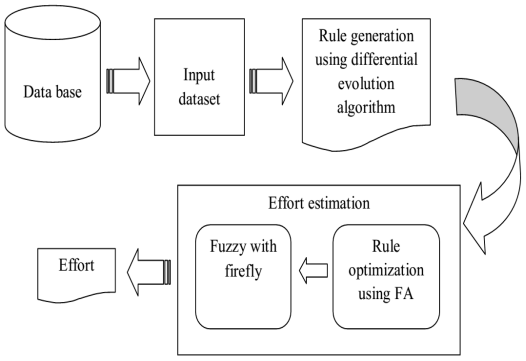

Fig. 1: The block diagram of the proposed method

Differential evolution algorithm: The DE technique, in turn, characterizes a population based approach such as the genetic algorithms making use of the identical operators such as the crossover, mutation and selection. The vital differentiation in configuring the superior solutions relies on the fact that the genetic algorithms are invariably dependent on the crossover while DE is basically based on the mutation function. The major task invariably depends on the divergences of arbitrarily sampled couples of solutions in the population.

The novel technique employs the mutation function as a search mechanism and the selection function to manage the search toward the potential zones in the search space. Further, it utilizes a non-uniform crossover which is capable of taking the child vector parameters from one parent more frequently than in the case from others. By means of the segments of the current population members to configure trial vectors, the recombination (crossover) operator effectively shuffles the data on thriving combinations, thereby leading the search towards a superlative solution space. The vital involved in the innovative technique are furnished as follows:

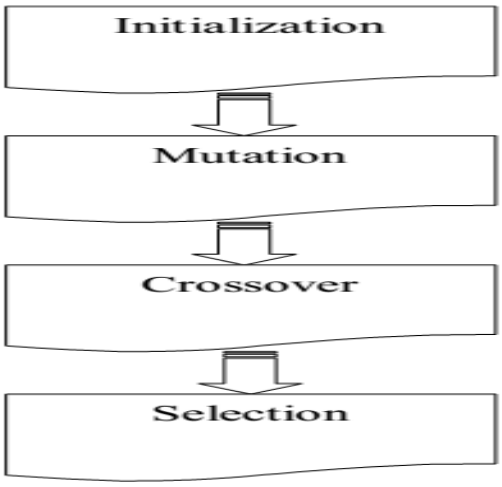

Fig. 2: Flowchart for the DE algorithm

- Initialization
- Mutation
- Crossover
- Selection

The flowchart for the differential evolution technique appears below in Fig.2.

Initialization: At the outset, let us suppose that the solution comprises n individuals. $S_{i,g}$ represents the ith individual of gth generation of the solution. At first, the initial solution is chosen arbitrarily.

$$S_{i,g} = \{s_1, s_2, ..s_n\} \quad (1)$$

Where, n corresponds to the number of individuals.

Mutation: There is a feast of various method intended for the mutation of individuals in differential evolution. As a rule, the mutation vector is generated as per Equation (2).

$$M_{i,g} + 1 = s_{i,g} + CF\left(s_{r1,g} - s_{i,g}\right) + SF\left(s_{r2,g} - s_{r3,g}\right) \quad (2)$$

where:
i and r1, r2, r3? {1,2,..n}
CF corresponds to the combination factor
SF signifies the scaling factor

Crossover: The parent vector is blended with the mutated vector to generate a fresh vector.

$$V_{ji,g+1} = \begin{cases} \text{if}\left(\text{rand} \le CR\right) \text{or} j = rn_i \\ S_{ji,g} \text{if}\left(\text{rand} > CR\right) \text{and} j \ne rn_i \end{cases} \quad (3)$$

Where:
j = 1, 2,..D;
Rand = [0, 1]
CR = Crossover rate [0, 1]
rn_i = (1,2,…D)

Selection: The entire solutions in the population enjoy an identical option of being shortlisted as the parents regardless of their fitness values. The child generated after the mutation and crossover functions is subjected to assessment. Subsequently, the performances of the child vector and its parent are assessed and contrasted, thereby leading to the choice of the superior one out of them. In the event of the parent maintaining the superior position, it continues to sustain the population. By means of the innovative algorithm, it is easy to achieve the superlative rules and apply them as the input of the fuzzy analogy controller for evaluating the effort. Subsequently, they are harmonized with the help of the firefly algorithm which proves its mettle in considerably cutting back the MMRE fault, thereby ushering in eye-catching outcomes for achieving the effort and the attendant expenses of the software product. The effort estimation procedure is discussed in a nutshell in the upcoming pages.

Effort estimation using fuzzy analogy with firefly algorithm: The Fuzzy logic is a fantastic method, well-geared to arrive at appropriate solutions to the vexed issues saddled with complications which cannot be comprehended quantitatively, through the fuzzy set theory. In fact, a fuzzy set can be explained scientifically by allotting to each potential individual in the universe of discourse a value characterizing its rank of membership in the fuzzy set to a bigger or smaller level as revealed by a greater or lesser membership grade. A fuzzy set represents a membership function, which correlates with each point in the fuzzy set an authentic number in the interval [0, 1], termed a degree or grade of membership. The membership function can be categorized into three types such as the triangular, trapezoidal and Gaussian. In our document, we opt for the triangular membership function for the purpose of effort estimation. The roadmap of the Fuzzy logic involves three phases as shown below:

- Fuzzification
- Fuzzy Rule-Based System
- Defuzzification
- Stage 1: Fuzzification: In this stage, a crisp input is modified into a fuzzy set
- Stage 2: Fuzzy Rule-Based System: In this system, 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 phase, the fuzzy output is converted into the crisp output

Fuzzy analogy: The Fuzzy Analogy 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 underlying motive behind this stage relates to the classification of the entire software projects by means of a set of attributes. Each software project is defined by a set of chosen attributes which are determined by linguistic values. For example, let us assume that there are M attributes and for each attribute (M_i) determine the linguistic variable (L_v). Each linguistic variable is characterized by a fuzzy set with the membership function (M_f). The fuzzy set and their membership functions are described with the help of automated and pragmatic methods. In the case of automated approaches the membership function is configured from the historical data. On the other hand, pragmatic approach arrives at the membership function through specialist knowledge. In our innovative technique, we follow the generation of the membership function from the historical data. Another appealing aspect about the amazing fuzzy analogy approach is that it duly takes into account the importance of each shortlisted attribute in the case detection. With the help of the above procedure, the rules are created and furnished as the input of the firefly algorithm. Given below is a description of the gradual procedure for the innovative technique.

Rule optimization by firefly algorithm: In the innovate technique, we deploy the firefly algorithm to adapt the created rules. Here, each solution is arbitrarily arrived at within a definite search space. Further each solution is characterized as rules P_{ix}, where: $P_{ix}= \{p_{i1} p_{i2}..p_{iy}\}$. At first, the fitness value of each rule is estimated. The rules which usher in the superlative fitness values are shortlisted as the current best rules. Subsequently, a ranking of the rules is performed in accordance with the fitness value:

$$\text{fitness} = \min \sum_{x=1}^{y} \text{MMRE} \tag{4}$$

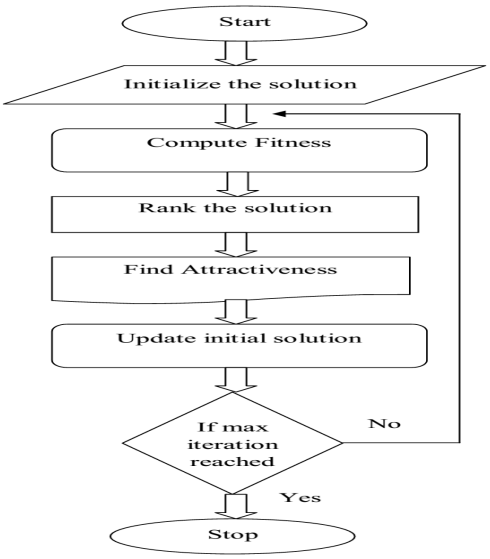

Fig. 3: The flowchart for firefly algorithm

We estimate the fitness values of the rules. The fitness value of each ith rule is analyzed and contrasted with the jth neighboring rule. If the fitness value of neighboring rule is found superior, we estimate the distance between every rule by means of the Euclidean distance measure. Now, the distance is employed to estimate the attractiveness (A):

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

Where:

A_0 = Refers to the preset attractiveness
γ = Represents the light absorption coefficient
d_{ij} = Denotes the distance between ith rule and jth neighboring rule

The new attractiveness value is employed to update the original rules with the help of Eq. 6:

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

Here, α and δ denote the evenly distributed values between 0-1. Therefore, the updated attractiveness values encourage the rule to inch towards the current best fitness value. If the fitness value of the new rule is superior to the original, it gets substituted by the new rule. Or else, the original rule continues to be in the population for the subsequent iteration. The total process gets repeated till the satisfaction of the stopping benchmark. The entire process is elegantly exhibited in the flowchart Fig.3.

Step 2: retrieval of similar cases: This phase is dependent on the selection of a software project similarity measure. In fact, the choice assumes greater significance, as it is bound to affect the fact regarding the analogies which are located. We have envisaged a set of candidate measures for software project resemblance. Let us assume two projects w1 and w2. We are competent to assess the overall resemblances of the two projects by integrating the individual resemblances of w1 and w2 linked with several attributes L_j.

Step 3: case adaptation: The underlying motive behind this phase is to develop an estimate for the new project by means of the identified effort values of identical projects. Here, we are faced with tackling of two vital challenges. The former relates to the selection of the number of identical projects to be utilized in the adaptation, whereas the latter lends itself the task of deciding the method to customize the shortlisted analogies so as to usher in an estimate for the new project. Each historical project contributes its mite in the evaluation of the effort of the new project, based on the level of resemblance with the related project.

Cost estimation: Software cost estimation is important cost estimation in software engineering that has become even more important in business software development due to changing requirements (Malathi and Sridhar, 2012). 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 DE algorithm as well as the Fuzzy analogy with FA technique.

RESULTS AND DISCUSSION

The proposed method is implemented in JAVA. The dataset used in the study is NASA 60. The NASA 60 dataset comprises of 60 complete projects Table 1 having

Table 1: Effort comparison of NASA 60 datasets

No of project	Actual effort	Estimated effort
20	295.25	309.23
40	215.19	221.113
60	406.413	410.448

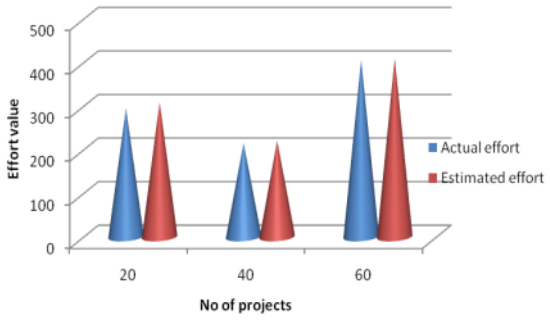

Fig. 4: Effort comparison of NASA dataset

17 independent variables of which 15 are categorical. The effort values obtained in NASA 60 dataset is described below.

In Fig. 4 demonstrates the effort comparison of number of projects for NASA 60 dataset. If the number of project is 20 the actual effort of the dataset is 295.25 and the estimated output of the implemented method is achieves 309.23. Then the number of project for the NASA 60 dataset is taken as 40, the actual effort value is 215.19 and the estimated effort is 221.113. If the number of project is 60 the actual effort of the NASA 60 data set is 406.413 and it achieves the estimated effort is 410.448.

Performance analysis: To measure the accuracy of the estimated effort generated by the proposed method we use the following metrics. Common method for evaluate the effort estimation is Mean Absolute Relative Error (MARE), Mean Magnitude of Relative Error (MMRE) and Prediction (PRED).

Mean Absolute Eelative Error (MARE):

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|act_{effort} - est_{effort}|}{act_{effort}} \quad (7)$$

Where: n = Is the number of projects

Mean Magnitude of Relative Error (MMRE): The MMRE can be measure by the following formula:

$$MMRE = \frac{1}{n}\sum_{i}^{n} MRE_i \quad (8)$$

Table 2: The MARE, MMRE and PRED measures of NASA 60 datasets

No of projects	MARE (%)	MMRE (%)	PRED (%)
20	0.0636	0.0426	45
40	0.0658	0.0448	72.5
60	0.04140	0.0273	88.33

Table 3: Execution time and memory value of proposed method

No of projects	Execution time	memory
20	5735	6509712
40	6849	6803056
60	5703	3864896

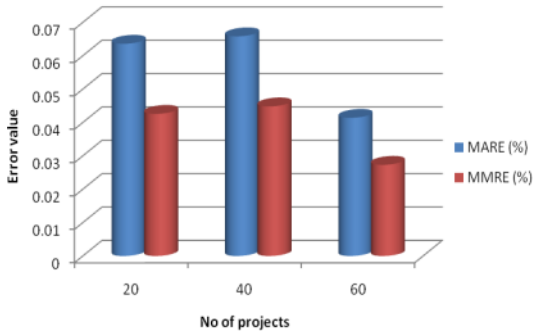

Fig. 5: MARE and MMRE measures of NASA 60 datasets

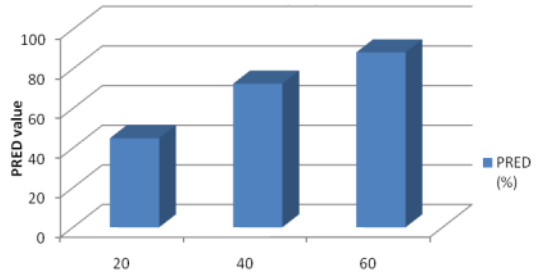

Fig. 6: PRED measures of NASA 60 the datasets

Prediction (pred):

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

n = Number of observation

The MARE, MMRE and PRED measures for the NASA 60 dataset on effort estimation process is described in Table 2 and Fig 5,6 It is tabulated in the below section:

Table. 3 is tabulating the execution time and memory value of the implemented method with different number of projects for NASA 60. The execution time and memory value is plotted in Fig.7 and Fig.8. It is shown in below section:

$$\text{PRED}(25) = \frac{k}{n} \qquad (9)$$

Comparative analysis for our proposed work with the existing works: The comparative analysis of our proposed method is compared with the various existing method is tabulated and the result are plotted given below: in Table 4

Table 4: comparison result of our proposed method

Metrics (%)	Existing method (Idri *et al.*, 2002) Malathi and Sridhar, 2012)	Proposed method
MMRE	0.26	0.03825
PRED	67	68.61
MARE	0.0988	0.05694

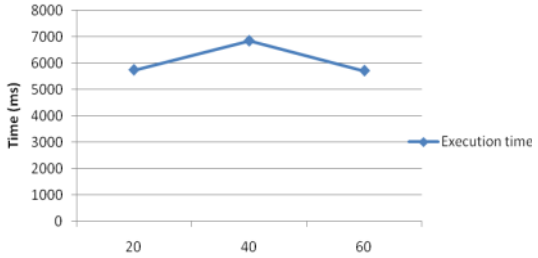

Fig. 7: Execution time of proposed method

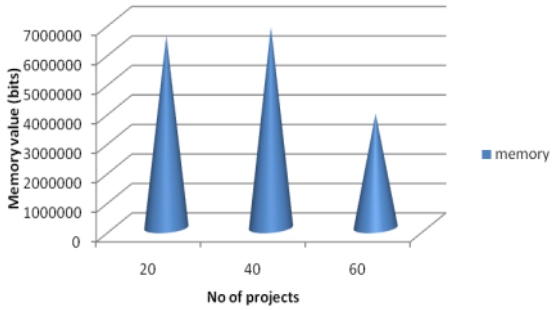

Fig. 8: Memory value of proposed method

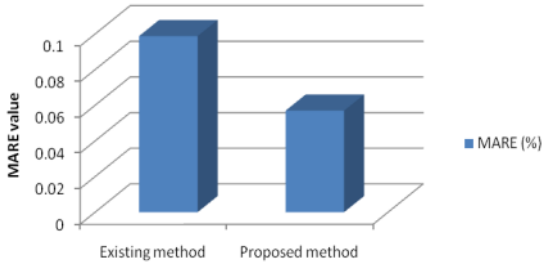

Fig. 9: Comparison value of proposed MARE with exisiting method

The PRED measure of the proposed method is described in Table 4 and the comparison of the PRED measure with existing method is described in this study.

The comparison of proposed method with the existing method based on the MMRE and MARE measure is plotted in Fig. 9 and 10. Here, the MMRE and MARE measure of NASA 60 dataset of the proposed is minimum error value when compared to the existing method.

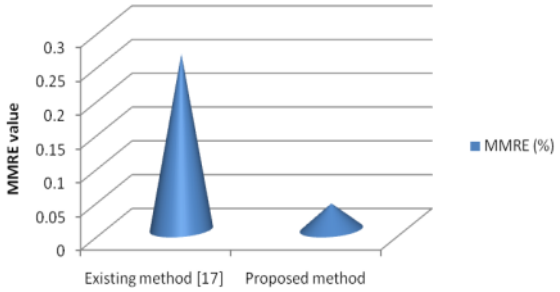

Fig. 10: Comparison value of proposed MMRE with exisiting method

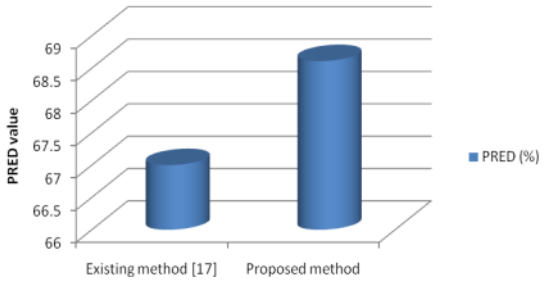

Fig. 11: Comparison value of proposed PRED with exisiting method

The proposed software effort estimation model shows that, it achieves lower MMRE, MARE and higher PRED (25%) (Fig. 11). The estimated efforts are more accurate than other models. Thus, the proposed method of cost estimation system based on soft computing technique is effectively estimate the effort and cost of the software.

CONCLUSION

In this study, the innovative hybrid method based fuzzy with firefly and differential evolution algorithm is elegantly launched for evaluating the effort of the software product. The epoch-making technique is performed in the JAVA platform. For producing the rules for the dataset is carried out by differential evolution algorithm. Now, the hybrid fuzzy with firefly algorithm is used to evaluate the effort with optimal rules. The Firefly algorithm discharges the task of harmonizing the rule with least error rate. Thus, we are gladdened to note that our charismatic technique comes out with flying colors in realizing superb outcomes vis-à-vis the parallel technique. It is hoped that the upcoming investigator will have ample opportunities to carry out their platform with their own unique optimization approaches so as to usher in superb outcomes.

REFERENCES

Al Dallal, J., 2010. Mathematical validation of object-oriented class cohesion metrics. Int. J. Comput., 4: 45-52.

Alsmadi, I. and H. Najadat, 2011. Evaluating the change of software fault behavior with dataset attributes based on categorical correlation. Adv. Eng. Software, 42: 535-546.

Attarzadeh, I. and S.H. Ow, 2010a. A novel soft computing model to increase the accuracy of software development cost estimation. Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE) 2010, February 26-28, 2010, IEEE, Singapore, ISBN: 978-1-4244-5586-7, pp: 603-607.

Attarzadeh, I. and S.H. Ow, 2010b. Proposing a new software cost estimation model based on artificial neural networks. Proceedings of the 2nd International Conference on Computer Engineering and Technology (ICCET) 2010, April 16-18, 2010, IEEE, Chengdu, China, ISBN: 978-1-4244-6347-3, pp: 487-491.

Azzeh, M., D. Neagu and P.I. Cowling, 2011. Analogy-based software effort estimation using Fuzzy numbers. J. Syst. Software, 84: 270-284.

Hari, C.H., R.P. Prasad, M. Jagadeesh and G.S. Ganesh, 2010. Interval type-2 fuzzy logic for software cost estimation using TSFC with mean and standard deviation. Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom) 2010, October 16-17, 2010, IEEE, Kottayam, India, ISBN: 978-0-7695-4201-0, pp: 40-44.

Idri, A., T.M. Khoshgoftaar and A. Abran, 2002. Can neural networks be easily interpreted in software cost estimation?. Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, May 12-17, 2002, IEEE, Honolulu, Hawaii, ISBN: 0-7803-7280-8, pp: 1162-1167.

Kad, S. and V. Chopra, 2012. Software development effort estimation using soft computing. Int. J. Mach. Learn. Comput., Vol. 2,

Khatibi, B.V., D.N. Jawawi, S.Z. Hashim and E. Khatibi, 2012. Increasing the accuracy of software development effort estimation using projects clustering. Software IET., 6: 461-473.

Malathi, S. and S. Sridhar, 2012. A novel approach to estimate the software effort based on fuzann technique. Eur. J. Sci. Res., 81: 563-574.

Merugu, R.R.R. and V.R.K. Dammu, 2012. Effort estimation of software project. Int. J. Adv. Res. Comput. Eng. Technol., 1: 33-41.

Orsila, H., J. Geldenhuys, A. Ruokonen and I. Hammouda, 2008. Update Propagation Practices in Highly Reusable Open Source Components. In: Open Source Development, Communities and Quality. Russo, B. and E. Damiani (Eds.). Springer, Berlin, Germany, ISBN: 978-0-387-09684-1, pp: 159-170.

Sehra, S.K., Y.S. Brar, and N. Kaur, 2011. Soft computing techniques for software project effort estimation. Int. J. Adv. Comput. Math. Sci., 2: 160-167.

Singh, B.K. and A.K. Misra, 2012. An alternate soft computing approach for efforts estimation by enhancing constructive cost model in evaluation method. Int. J. Innovation Manage. Technol., Vol. 3.

Yadav, R.K. and S. Niranjan, 2013. Software effort estimation using fuzzy logic: A review. Intl. J. Eng. Res. Technol., 2: 1377-1384.

Zia, Z. and A. Rashid, 2011. Software cost estimation for component based fourth-generation-language software applications. Software IET., 5: 103-110.

Ziauddin, S.K.T., K. Zaman and S. Zia, 2012. Software cost estimation using soft computing techniques. Adv. Inf. T echnol. Manage., 2: 233-238.