Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 143 (2018) 325–332

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Computing and Communication (ICACC-2018)

Software Cost Estimation Using Environmental Adaptation Method

Tribhuvan Singh[a,*], Ranvijay Singh[a], Krishn Kumar Mishra[a]

[a]*Motilal Nehru National Institute of Technology Allahabad, Allahabad-211004, India*

Abstract

Environmental adaptation method (EAM) is one of the evolutionary algorithms for solving single objective optimization problems. After the first proposal of EAM, other variants have been suggested to speed up the convergence and to maintain the population diversity. Among them, IEAM-RP works with real numbers and was able to achieve the desired goal during the optimization process. In this paper, IEAM-RP is used to predict the effort required to develop the software product. The experiments are carried out on NASA software project dataset to check the effectiveness of IEAM-RP. The experimental results demonstrated that the overall performance of IEAM-RP is quite satisfactory in predicting the effort required to develop a software.

Keywords: Software Cost Estimation; Evolutionary Algorithms; Optimization Problems; Environmental Adaptation Method.

1. INTRODUCTION

In the last three decades, many software cost estimation models have been suggested. This area is so significant that it has gained continuous research attention. Evaluating the project estimated cost, duration, and maintenance cost during the software development is a valuable goal that needs to be achieved to reduce overall development cost. The primary element that affects the estimated efforts is development line of code (DLOC). Boehm et al. [6], [5] suggested one of the famous model to estimate the software effort in an efficient way. This model is called as COnstructive COst MOdel (COCOMO). In this model, a total of 63 software projects were used to evaluate the performance.

In the literature, various techniques have been given for estimating the software development effort. These techniques involve soft computing, swarm-based algorithms, evolutionary algorithms, and many more. Among them, evolutionary algorithms were found successful in estimating the effort in an efficient way due to their population-based search technique. Shepperd et al. [22] suggested an analogy-based system to estimate the software effort. Kumar et al. [13] introduced a fuzzy logic and neural network-based model for software cost estimation. Kaczmarek et al. [11] introduced a model for size and effort estimation for applications written in Java. Jeffery et al. [10] suggested a soft-

* Tribhuvan Singh
E-mail address: 2015rcs09@mnnit.ac.in

10.1016/j.procs.2018.10.403

ware cost estimation model using public domain metrics. In recent years, some multiobjective techniques have also been suggested that optimize minimum two conflicting objectives (RMSE, MMRE etc.) simultaneously of cost estimation model. In addition to the application of cost estimation, some other applications of multiobjective techniques are surveyed in [14].

Recently a new evolutionary algorithm IEAM-RP [25] has been suggested for solving single objective optimization problems. The performance of the suggested algorithm was checked using 24 benchmark functions of COCO [9] framework. The performance of IEAM-RP was found quite satisfactory in terms of convergence rate and population diversity. IEAM-RP is also used to compute modularity value in [19]. In this paper, IEAM-RP is used to tune the parameters (a, b, c, d) of Sheta model for which MRE and MMRE are optimized as compared to existing models.

The rest of the paper is organized as follows. Section II describes the related work and background details. The description of the fitness function and evaluation metrics have been given in Section III. Section IV describes the basic idea of the suggested model for software cost estimation. The analysis of experimental results is given in Section V. Section VI concludes the paper and highlights the future direction.

2. RELATED WORK AND BACKGROUND DETAILS

2.1. Related work of software cost estimation

Boehm et al. [6] introduced a non-linear mode of assessment method in 1981, called as COCOMO (COnstructive COst MOdel). This model was divided into three different levels, namely, basic model, intermediate model, and detailed model. This model is widely used to evaluate the software effort. Other techniques have also been suggested in the past for estimating the effort of a software project. Soft computing methods were explored to build efficient effort estimation models structures. Kelly [12] utilized the concept of neural networks, genetic algorithms, and genetic programming to introduce a methodology for software cost estimation. Later, Dolado et al. [7] provided a detailed study on using neural networks, genetic programming, and linear regression in solving the project effort estimation. In [1], [17], many datasets were given for software cost estimation. Kumar et al. [13] introduced a fuzzy logic and neural network-based software cost estimation model.

Recently, machine learning and soft computing techniques were explored to handle effort and cost estimation problems. In [23], the author provided an approach for a different set of models modified from the COCOMO model using genetic algorithms. After the proposal of [23], many authors explored the same idea with some modification [20], [26], [21]. Authors of [20], [26], [21] have provided a comparison of results to the work presented in [23]. Nature-inspired algorithms were found more suitable for solving software cost estimation problems due to their population-based search technique. Many nature-inspired algorithms were suggested for cost estimation of a software project in recent years. Mishra et al. [18] introduced a memetic algorithm for software cost estimation. Authors of [27] suggested a genetic algorithm-based parameter tuning approach for COCOMO II model for estimating software development effort. Lin et al. [16] introduced a particle swarm optimization-based search technique to estimate software effort. Aljahdali et al. [2] proposed a differential evolution-based software cost estimation technique for tuning the COCOMO model parameters.

2.2. Introduction to Improved Environmental Adaptation Method with Real Parameter (IEAM-RP)

The IEAM-RP [25] is one of the evolutionary algorithms that is used to solve single objective optimization problems. This algorithm is based on the principle of adaptive learning developed by J. M. Baldwin [4]. This algorithm achieves the principle of adaptive learning using its operators, adaptation, and selection. Adaptation operator basically receives parent population and creates offspring. In order to create offspring, adaptation operator divides the population into two classes. One class contains only one (best) solution, whereas another class contains remaining solutions. The best solution updates its position vector using its personal fitness and average fitness of the population as follows:

$$P_{b+1} = P_b \times F(P_b)/F_{avg} + \beta \tag{1}$$

Here, β is a random value between 0 to 1. P_b and P_{b+1} are the old and updated position vectors of the best solution. $F(P_b)$ is the fitness value of P_b and F_{avg} is the average fitness of the population. Here, the symbol / is used for division operator. The remaining solutions receive the direction from the best solution and try to achieve a better phenotypic structure in the problem search space as follows:

$$P_{nb+1} = P_{nb} + \beta \times (\text{best\_position} - \text{worst\_position}) \tag{2}$$

Here, P_{nb}, and P_{nb+1} are the old and updated position vectors of the non-best solutions. In minimization problems, a solution with least fitness is called as the best solution, whereas a solution with maximum fitness is called as the worst solution. The difference of the best_position to the worst_position is the positional bandwidth of the best solution to the worst solution. The non-best solutions utilize this bandwidth to exploit in the problem search space. The left-hand side of equation 1, and 2 are the offspring corresponding to the best, and non-best solutions, respectively. In order to get the offspring corresponding to the population, we need to combine left-hand side of equations 1, and 2. The selection operator combines parent and offspring to get N best solutions. Here, N is the size of the initial population.

3. Description of fitness functions and evaluation metrics

3.1. Mathematical Formulation of Evolution Techniques

In this paper, MRE, MMRE, and PRED (L) [23], [15], [8], [24] are used for cost estimation model. Here, MRE, and MMRE are taken as fitness functions that need to be minimized. The mathematical expressions of these measurement techniques are given below.

3.1.1. Relative Error (RE)

This error is one of the measurement technique that is used to measure the quality of a model. This value should be minimum for a better estimation model. The RE can be measured as follows:

$$RE = \frac{\text{measured effort} - \text{estimated effort}}{\text{measured effort}} \tag{3}$$

3.1.2. Magnitude of Relative Error (MRE)

The absolute value of RE of the respective project is called as the magnitude of relative error, computed as follows:

$$MRE_i = \text{abs}\frac{\text{measured effort}_i - \text{estimated effort}_i}{\text{measured effort}_i} \tag{4}$$

3.1.3. Mean Magnitude of Relative Error (MMRE)

MMRE is the average of MRE over n number of observations. Mathematically, MMRE can be represented as follows:

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n} \text{MRE}_i \tag{5}$$

3.1.4. Prediction at Some Level (PRED)

This measurement technique evaluates the performance of an estimation model. The PRED with prediction at level L can be computed as follows:

$$\text{PRED (L)} = \text{k/n} \tag{6}$$

where L is the limit for k, whereas k represents the total number of observations less than or equal to L. Here, n is the total number of observations. In this paper, the value of L is generally taken as 0.25 for the measurement. The minimum value of MMRE and the maximum value of PRED is desirable for an estimation model.

3.2. Software Cost Estimation Models

Sheta proposed evolutionary models using genetic algorithms (GAs) for estimating the software effort [23]. He applied GA to estimate the parameters of COCOMO effort estimation model. The performance of the proposed models was tested on 18 software project dataset taken from NASA [3]. The magnitude of relative error (MRE) and mean magnitude of relative error (MMRE) were considered as the fitness functions to evaluate the performance of the cost estimation models. Sheta has given three models to calculate estimated efforts that are given below.

Model 1: This model uses two parameters, a, and b with a developed line of code (DLOC). The estimated effort (EE) for this model can be computed as follows:

$$EE = a \times (DLOC)^b \tag{7}$$

Model 2: This model uses three parameters, a, b, and c with DLOC, and methodology (M) to compute estimated effort. Here, methodology is used to improve the prediction capability of the basic COCOMO model. The value of methodology for each software project is given in table 1. The estimated effort is computed as follows:

$$EE = a \times (DLOC)^b + c \times (M) \tag{8}$$

Model 3: This model uses four parameters (a, b, c, d) with DLOC and methodology to compute estimated effort as follows:

$$EE = a \times (DLOC)^b + c \times (M) + d \tag{9}$$

DLOC and M are given, the parameters are optimized in such a way that MRE and MMRE should be minimized. Boundary range of parameters of Sheta model is given below.
Search Domain for a {0, 10}
Search Domain for b {0.3, 2}
Search Domain for c {-0.5, 0.5}
Search Domain for d {0, 20}.
The objective of the proposed technique is to tune the above parameters that provide most accuracy in cost estimation.

Table 1: Description of NASA Software Project Data

Project No.	KDLOC	Methodology (M)	Measured Effort
1	90.2000	30.0000	115.8000
2	46.2000	20.0000	96.0000
3	46.5000	19.0000	79.0000
4	54.5000	20.0000	90.8000
5	31.1000	35.0000	39.6000
6	67.5000	29.0000	98.4000
7	12.8000	26.0000	18.9000
8	10.5000	34.0000	10.3000
9	21.5000	31.0000	28.5000
10	3.1000	26.0000	7.0000
11	4.2000	19.0000	9.0000
12	7.8000	31.0000	7.3000
13	2.1000	28.0000	5.0000
14	5.0000	29.0000	8.4000
15	78.6000	35.0000	98.7000
16	9.7000	27.0000	15.6000
17	12.5000	27.0000	23.9000
18	100.8000	34.0000	138.3000

4. DESCRIPTION OF THE ALGORITHM

In this paper, IEAM-RP is used for tuning the parameters of Sheta model for software effort estimation. The algorithm starts with a randomly initialized population of the specified size in the boundary range. Each individual of the population has four parameters (a, b, c, d). These parameters need to be estimated in such a way that the final MMRE value should be minimized. Random initialization of the individuals gives a matrix of size N×D. Here, N is the population size and D is the dimension size of the problem taken under consideration. In this paper, N = 18 and D = 4. After random initialization of the population, parametric values of the first row are used to compute EE and MRE for all software projects. Afterward, the average of all MREs i.e. MMRE is computed. The value of MMRE and the corresponding values of a, b, c, and d are stored in a temporary memory. This process will be continued for rest of the row vectors, and MMRE, four parametric values are computed. Finally, the minimum value of MMRE and the corresponding value of four parametric values are saved in temporary variables. After first generation, the four parametric values are updated using equations 1 and 2. These updated values make the offspring of size N for parent population. The parent population and offspring are combined and N best individuals are selected for the next generation. Now, these individuals are used to create offspring in the same way as discussed above. At the end of the program execution, the minimized value of MMRE and the corresponding value of parameters (a, b, c, d) are stored. Finally, the optimum value of MMRE and the corresponding value of four parameters will be obtained. The steps of IEAM-RP for tuning the parameters of software cost estimation problem are given below.

Step 1: Initialize the value of population size as 18 and maximum number of generations as 100. Create initial population randomly.

Step 2: While (number of generations < MaxGen)

Repeat the following steps 3 to 8 until the specified number of generations (MaxGen) has not reached.

Step 3: For each row vector of parameter values in the matrix do

for i=1, ... ,PS

Evaluate the value of MRE for all software projects using equation 4.

Evaluate the value of $MMRE_i$ of all MREs obtained using equation 5. Here, $MMRE_i$ is the mean value of MREs of all software projects for i^{th} individual of the population.

end for

Step 4: Store minimum MMRE value and corresponding parameters (a, b, c, d) values in temporary variables.

Table 2: Optimized value of parameters by EAMDGA and IEAM-RP for model 1

Parameters	EAMDGA	IEAM-RP
a	1.3889	1.9512
b	0.9889	0.907

Table 3: Comparison of MMRE and PRED for cost estimation model 1

Evaluation Metrics	Sheta Model 1	Sharma et al.	EAMDGA	IEAM-RP
MMRE (%)	23.79	23.25	19.07	18.84
PRED (0.25)	39.89	62.34	73.13	72.42

Table 4: Optimized value of parameters by EAMDGA and IEAM-RP for model 2

Parameters	EAMDGA	IEAM-RP
a	0.569	1.0055
b	1.2012	1.0602
c	0.12891	0.0997

Step 5: Apply adaptation operator to create offspring.
Step 6: Initial/Intermediate population and offspring are combined.
Step 7: Apply selection operator on combined population to select N best solutions.
Step 8: Number of generations = number of generations + 1.
Step 9: End while.
Step 10: Select minimum MMRE with corresponding value of four parameters.
Step 11: Print the final values of a, b, c, d, and MMRE.

After termination of the above algorithm, the optimized value of MMRE with corresponding values of four parameters are obtained. On the basis of four parameters, the values of EE, and MRE can be calculated.

5. ANALYSIS OF EXPERIMENTAL RESULTS

In this section, the experimental results and analysis have been given. The performance of IEAM-RP is analyzed on the basis of MMRE and PRED values. The estimated effort and MMRE obtained by IEAM-RP gives better results compared to the existing technique. The optimized values of parameters by EAMDGA and IEAM-RP of Sheta model are given in tables 2, 4, 6 for model 1, 2, and 3, respectively. The optimized values of MMRE and PRED for all algorithms are given in tables 3, 5, 7 for model 1, 2, and 3, respectively. From table 2, it is easy to notice that the minimum value of a, and b are given by EAMDGA, and IEAM-RP, respectively. Table 3 indicates better values of MMRE, and PRED for model 1 are given by IEAM-RP, and EAMDGA, respectively. The optimized values of parameters for model 2 are given in table 4. This table shows that minimum values of a, b, and c are given by EAMDGA, IEAM-RP, and IEAM-RP, respectively. The best value of MMRE for model 2 is given by IEAM-RP. The reason is optimized values of parameters are more suitable that are able to minimize the difference between measured effort and estimated effort. The optimized values of parameters of model 3 by EAMDGA and IEAM-RP are given in table 6. The best values of MMRE and PRED for model 3 are given by IEAM-RP, and EAMDGA, respectively.

The detailed results analysis has shown that the experimental values produced by IEAM-RP are very close to the standard values of the models. The second best values of the metrics are obtained by EAMDGA. The reason behind the competitive performance of IEAM-RP is due to the fact that its convergence speed and population diversity is better compared to other existing algorithms.

Table 5: Comparison of MMRE and PRED for cost estimation model 2

Measurement Techniques	Algorithm Name Sheta Model 1	Sharma et al.	EAMDGA	IEAM-RP
MMRE (%)	22.61	20.56	19.7	17.49
PRED (0.25)	61.11	66.67	72.22	72.29

Table 6: Optimized value of parameters by EAMDGA and IEAM-RP for model 3

Parameters	EAMDGA	IEAM-RP
a	0.4993	1.0818
b	1.2312	1.0522
c	0.09785	-0.4573
d	1.2457	15.3708

Table 7: Comparison of MMRE and PRED for cost estimation model 3

Measurement Techniques	Algorithm Name Sheta Model 1	Sharma et al.	EAMDGA	IEAM-RP
MMRE (%)	63.64	20.33	19.63	11.74
PRED (0.25)	38.39	72.22	77.78	75.84

6. CONCLUSION

In this paper, IEAM-RP is used for tuning the parameters of Sheta model of software cost estimation. The results produced by IEAM-RP have shown its superiority compared to other existing techniques. The reason behind the competitive performance of IEAM-RP is its convergence speed and diversity preservation capabilities of the solutions during the optimization process. This algorithm is able to minimize the difference between measured effort and estimated effort. The best result of MMRE is produced by IEAM-RP for all three models. The best prediction was given by EAMDGA, IEAM-RP, and EAMDGA for model 1, 2, and 3, respectively. Extensive experimental results and analysis demonstrated the effectiveness of the proposed technique.

References

[1] A. J. Albrecht and J. E. Gaffney. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE transactions on software engineering*, (6):639–648, 1983.

[2] S. Aljahdali and A. F. Sheta. Software effort estimation by tuning coocmo model parameters using differential evolution. In *Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on*, pages 1–6. IEEE, 2010.

[3] J. W. Bailey and V. R. Basili. A meta-model for software development resource expenditures. In *Proceedings of the 5th international conference on Software engineering*, pages 107–116. IEEE Press, 1981.

[4] J. M. Baldwin. A new factor in evolution (continued). *The American Naturalist*, 30(355):536–553, 1896.

[5] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby. Cost models for future software life cycle processes: Cocomo 2.0. *Annals of software engineering*, 1(1):57–94, 1995.

[6] B. W. Boehm et al. *Software engineering economics*, volume 197. Prentice-hall Englewood Cliffs (NJ), 1981.

[7] J. J. Dolado and L. Fernandez. Genetic programming, neural networks and linear regression in software project estimation. In *Procs. of International Conference on Software Process Improvement, Research, Education and Training*, pages 157–171, 1998.

[8] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29(11):985–995, 2003.

[9] N. Hansen, A. Auger, S. Finck, and R. Ros. *Real-parameter black-box optimization benchmarking 2010: Experimental setup*. PhD thesis, INRIA, 2010.

[10] R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International*, pages 16–27. IEEE, 2001.

[11] J. Kaczmarek and M. Kucharski. Size and effort estimation for applications written in java. *Information and Software Technology*, 46(9): 589–601, 2004.

[12] M. A. Kelly. A methodology for software cost estimation using machine learning techniques. Technical report, NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 1993.

[13] S. Kumar, B. A. Krishna, and P. S. Satsangi. Fuzzy systems and neural networks in software engineering project management. *Applied Intelligence*, 4(1):31–52, 1994.

[14] S. Lalwani, S. Singhal, R. Kumar, and N. Gupta. A comprehensive survey: Applications of multi-objective particle swarm optimization (mopso) algorithm. *Transactions on Combinatorics*, 2(1):39–101, 2013.

[15] Y.-F. Li, M. Xie, and T. N. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82(2):241–252, 2009.

[16] J.-C. Lin and H.-Y. Tzeng. Applying particle swarm optimization to estimate software effort by multiple factors software project clustering. In *Computer Symposium (ICS), 2010 International*, pages 1039–1044. IEEE, 2010.

[17] J. E. Matson, B. E. Barrett, and J. M. Mellichamp. Software development cost estimation using function points. *IEEE Transactions on Software Engineering*, 20(4):275–287, 1994.

[18] K. Mishra, A. Tripathi, S. Tiwari, and N. Saxena. Evolution based memetic algorithm and its application in software cost estimation. *Journal of Intelligent & Fuzzy Systems*, 32(3):2485–2498, 2017.

[19] K. K. Mishra, H. Bisht, T. Singh, and V. Chang. A direction aware particle swarm optimization with sensitive swarm leader. *Big Data Research*, 2018.

[20] H. Mittal and P. Bhatia. A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers. *International Journal of Computer Science and Security*, 1(4):36–47, 2007.

[21] P. S. Sandhu, M. Prashar, P. Bassi, and A. Bisht. A model for estimation of efforts in development of software systems. *World Academy of Science, Engineering and Technology*, 56:148–152, 2009.

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

[23] A. F. Sheta. Estimation of the cocomo model parameters using genetic algorithms for nasa software projects. *Journal of Computer Science*, 2 (2):118–123, 2006.

[24] B. K. Singh, A. Punhani, G. P. Singh, and A. Misra. Effect of variations in measurement process for software development efforts. In *Multimedia, Computer Graphics and Broadcasting (MulGraB), 2014 6th International Conference on*, pages 3–8. IEEE, 2014.

[25] T. Singh, A. Shukla, and K. Mishra. Improved environmental adaption method with real parameter encoding for solving optimization problems. In *Advances in Computer and Computational Sciences*, pages 13–20. Springer, 2018.

[26] M. Uysal. Estimation of the effort component of the software projects using simulated annealing algorithm. *World Academy of Science, Engineering and Technology*, 41:258–261, 2008.

[27] C. S. Yadav and R. Singh. Tuning of cocomo ii model parameters for estimating software development effort using ga for promise project data set. *International Journal of Computer Applications*, 90(1), 2014.