EVOLUTIONARY COMPUTING TECHNIQUES FOR SOFTWARE EFFORT ESTIMATION

Sumeet Kaur Sehra[1], Yadwinder Singh Brar[1], Navdeep Kaur[2]

[1]I.K.G. Punjab Technical University, Jalandhar, Punjab, India
[2] Sri Guru Granth Sahib World University, Fatehgarh Sahib, Punjab, India

ABSTRACT

Reliable and accurate estimation of software has always been a matter of concern for industry and academia. Numerous estimation models have been proposed by researchers, but no model is suitable for all types of datasets and environments. Since the motive of estimation model is to minimize the gap between actual and estimated effort, the effort estimation process can be viewed as an optimization problem to tune the parameters. In this paper, evolutionary computing techniques, including, Bee colony optimization, Particle swarm optimization and Ant colony optimization have been employed to tune the parameters of COCOMO Model. The performance of these techniques has been analysed by established performance measure. The results obtained have been validated by using data of Interactive voice response (IVR) projects. Evolutionary techniques have been found to be more accurate than existing estimation models.

KEYWORDS

Machine Learning, COCOMO, MMRE, Evolutionary computing

1. INTRODUCTION

Software effort estimation (SEE) is the task of estimating the needed effort to develop a software. Complexity in estimating software development effort has been always attention area for researchers [2, 24]. Overestimation of effort may result in problems in project scheduling and faulty software, whereas underestimation of the effort may result in over budgeting and delayed delivery of system [21].

Various methods including, algorithmic models, namely COCOMO [18], expert judgement [19, 20] have been employed for developing SEE models. In recent years, machine learning (ML) techniques such as neural networks [3, 26] bagging predictors [6] and support vector regression (SVR) [10, 25] have been investigated to develop SEE models. Some researchers [14, 15] have considered ML based method as a major category of SEE methods. Machine Learning (ML) techniques learn and use the knowledge from the historical project to estimate the effort. Due to the extensive coverage of solution space, ML techniques have contributed to the software effort estimation models [7].

Evolutionary computing gets inspiration from natural biological evolution and adaptation. In recent years, evolutionary computing algorithms have been used in software effort estimation by various researchers [5, 8, 17, 22, 29, 30]. Although numerous models have been proposed still no model has been found to be highly accurate or efficient approach in all the environments. Since identification of an estimation method based upon historical data is viewed as a problem to minimize the difference between the actual and predicted effort. Thus, it can be seen as an optimization problem and evolutionary based approaches can be employed to solve this problem. In SEE optimization problem, the given evaluation criterion (usually a measure of predictive

DOI:10.5121/ijcsit.2017.9211

performance) is globally optimized [16]. In this article, optimization techniques, including, bee colony, particle swarm and ant colony have been employed to estimate the software effort.
The remainder paper is structured in five sections. The next section discusses the related research work. In third section, techniques used for optimization namely Particle Swarm Optimization, Ant Colony Optimization and Bee Colony Optimization are presented. Fourth section elaborates model experimentation and data set used for validation of the models. In fifth section, results are presented and compared with COCOMO model. Last section concludes the paper and provides the scope for future work.

2. RELATED WORK

EC based optimization algorithms motivated from nature have gained attention of researchers for generating more accurate estimates. In SEE research, various EC based algorithms have been employed and validated by researchers to formulate an estimation model for predicting the effort. Genetic Algorithms (GAs) have been hybridized with other approaches aiming to improve some existing model based techniques [16, 18, 31]. Aljahdali and Sheta [1] have developed a model based upon differential evolution COCOMO-DE and the performance of the developed model was tested on NASA software project dataset. The proposed model was able to provide good estimation capabilities. Chalotra et al. [8] have implemented Bee Colony Optimization (BCO), a meta-heuristic way to optimize the parameters of COCOMO. The results obtained show that the proposed BCO based model is able to improve the accuracy of cost estimation and also outperform other models. Chen and Zhang [9] have proposed an approach based on an event-based scheduler (EBS) and ant colony optimization (ACO) algorithm which enables the modelling of resource conflict and task preemption. Dewan and Sehra [11] have utilized ACO to optimize the parameters of the organic mode of Basic COCOMO model using ACO technique and have concluded that the proposed model was able to perform better than the COCOMO model. PSO has been implemented by [4, 22, 28] for modification of the COCOMO parameters and it has been concluded that the resultant techniques increase the accuracy of model and flexibility related to the software.

3. METHODOLOGY

3.1 Particle Swarm Optimization

Kennedy and Eberhart [23] proposed PSO algorithm being inspired by the birds'social behaviour. PSO is based on stochastic optimization technique, inspired from flocking birds and fish schooling. The algorithm is based upon the concept that particles of a swarm calculate their position based on fitness function from the problem space in which they move. The fitness function is specified on the basis of the problem to be optimized and is referred as f(Xi), short for f(Xi.O),. ..... ,f(Xi. d), in which i is the particle, and d is number of dimensions to be optimized.
In this algorithm, all the particles are completely connected, sharing the information about the best position. Position and velocity of each particle are calculated by equations 1 and 2 as follows:

$$Xi,d(it + 1) = Xi,d(it) + Vi,d(it + 1) \quad \ldots (1)$$

$$Vi,d(it + 1) = Vi,d(it) + C1 * Rnd(0, l) * [pbi,d(it) - Xi,d(it)] + C2 * Rnd(0, l) * [gbd(it) - Xi,d(it)] \quad \ldots(2)$$

Where i is particle id; d represents dimension being considered; it is iteration number; Xi,d is particle i's position; Vi,d is particle i's velocity; Cl and C2 are acceleration constants; Rnd is a random value; pbi,d is the particle best location; and gbd is the global best location.

3.2 Ant Colony Optimization

ACO algorithm was introduced by Dorigo and Blum [13]. Observation of ant colonies has inspired development of this algorithm. Ants' foraging behaviour is the basis of the algorithm, i.e. how shortest paths are searched by ants between food sources and their nests. Initially, ants initially search the surrounding area in a random manner. During this, a chemical pheromone trail is left on the ground by the ants. When ants choose the path, they choose the paths which have been marked with strong concentrations of pheromone. As soon as it finds a food source, it evaluates the quantity and quality of food while taking back some of the food. The pheromone trails during the return trip guides other ants to find the food source. Thus, shortest paths between nest and food source are identified based upon trail left on ground.

3.3. Bee Colony Optimization

The Bee Colony Optimization (BCO) is a perfect example of swarm intelligence [27]. The BCO approach is a "bottom up" modelling approach creating artificial agents analogous to bees. Artificial bee agents collaboratively solve complex combinatorial optimization problem. BCO is a meta-heuristic algorithm which uses the swarm behaviour of bees and collective intelligence to deal with combinatorial problems [12]. Bee Algorithm is used for finding the best possible solutions for optimization problems. Artificial bees are responsible for solving problems. Bee algorithm consists of two types of alternating pass that contribute in a single step, forward pass and backward pass. Both passes are problem dependent. During forward pass every bee is assigned with an empty solution. All bees explore the search space on an individual basis for number of predefined moves. Partial/complete solutions are computed by every bee. This evaluation depends on individual exploration and past experience. After that bees go back to hive or colony and start the second phase of first step i.e. backward pass. During this all bees participate in the decision making process and all evaluated solutions are shared by every bee by performing waggle dance which is in the shape of digit '8'. The solutions vary from bee to bee. This is the only time when bees communicate with one another and the best among all solution is considered as a partial/complete solution of a problem. Only those solutions will be loyal that will be giving best solutions. Equation 3 is used for selecting the best solution:

$$\text{Loyalty} = \frac{\text{maximum value - current value}}{\text{maximum value - minimum value}} \quad \ldots\ldots (3)$$

Where, max value is the maximum value from set of solutions, min value is the minimum value from set of solutions, and current value is the value for current solution. Loyalty will be checked for every bee solution within a single move then from those set of solutions above mentioned value will be taken.

4. MODEL EXPERIMENTATION

One of the foremost problems that arises in software development industry is to estimate the effort. Large number of models is available but very few reach the level of satisfaction. COCOMO Model is a very famous model that gives a level of satisfaction with positive results. But due to high demands and complex software's it has become less accurate. In this paper, various techniques including, ACO, BCO and PSO have been implemented to tune the parameters of COCOMO model. IVR dataset has been used for evaluation and validation of these techniques. IVR dataset consists of 48 projects along with its size (in KLOC), actual effort (in PM) and time duration (in months) [32]. The equation for evaluating the effort using COCOMO is shown in equation 4.

$$\text{Effort} = 2.4 * (KLOC)^{1.05} \quad \ldots\ldots. (4)$$

KLOC is a physical measure of size of source program that counts source lines and excludes comment and blank space in COCOMO model. The performance is evaluated by Equations 5 and 6.

$$\text{MMRE} = \frac{1}{N} \sum_{i=1}^{N} \frac{\text{actual effort- predicted effort}}{\text{actual effort}} \quad \ldots\ldots\ldots(5)$$

$$\text{RMSE} = \sqrt{\sum_{i=1}^{N} \frac{1}{N} (\text{actual effort- predicted effort})^2} \quad \ldots\ldots (6)$$

]

where actual effort is taken from the data set and predicted effort is the effort calculated using proposed technique.

5. RESULTS

The validation of implemented EC techniques is performed by using a dataset of Interactive Voice Response (IVR) applications from software industry [32]. The performance of different techniques is compared with actual effort of the projects as depicted in Table 1. From Table 1, it is evident that effort of the projects after applying different optimization algorithms is quite close to the actual effort of dataset [32]. Also from Table 2 and 3, the values of MMRE and RMSE for BCO is 0.11 and 7.85 respectively, minimum as compared to other optimization techniques. Figure 1a depicts the effort values estimated using COCOMO, ACO, BCO and PSO techniques. It is clear that effort estimated using EC techniques is relatively closer to actual effort form the dataset. Figures 1b and 1c present the comparison of RMSE and respectively and reveal that BCO outperforms relatively than other EC techniques applied.

Table 2. MMRE comparison

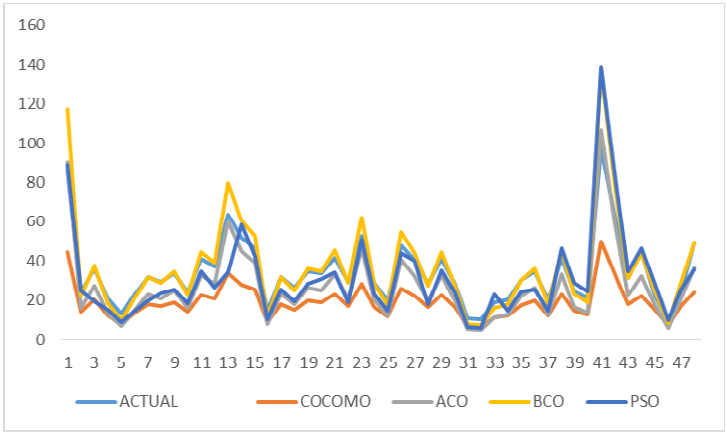
S.No	Model used	MMRE
1	COCOMO	0.43
2	ACO	0.27
3	BCO	0.11
4	PSO	0.19

Table 2. RMSE comparison

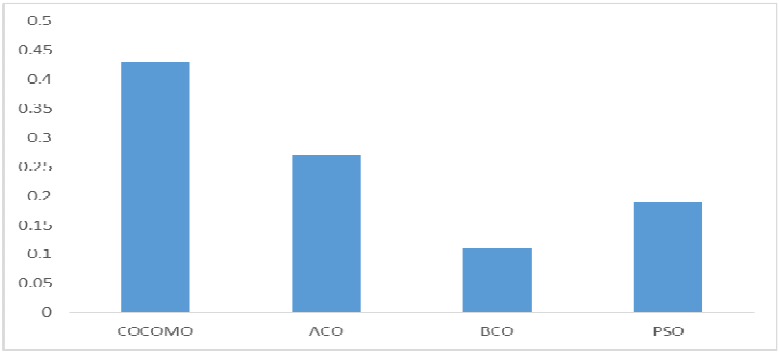
S.No	Model used	RMSE
1	COCOMO	17.49
2	ACO	8.57
3	BCO	7,85
4	PSO	10.26

6. CONCLUSION

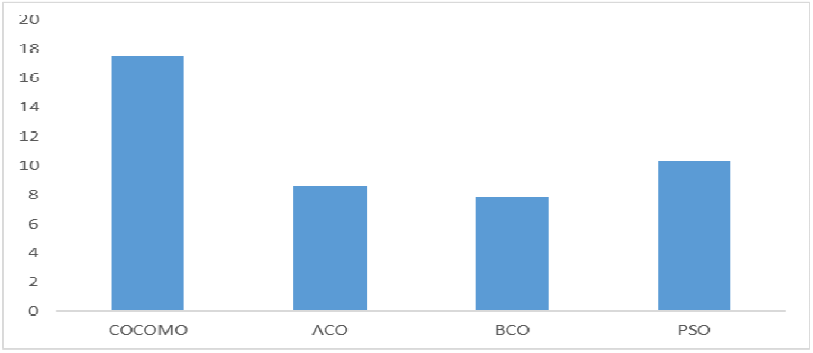
Accurately estimating the effort required to develop a software has always been a concern for industry and academia. EC based algorithms have been considered as efficient optimization algorithms which can be implemented on very few parameters. In this paper, EC techniques have been employed for SEE to provide better estimates. The performance of these techniques has been validated using 48 projects of IVR dataset. The value of MMRE for BCO is 0.11 which is comparatively less than other existing state of the art SEE algorithmic models. In future, these EC techniques can be used with other meta-heuristic techniques to develop ensembles for SEE and more datasets can be used to validate the model.



a) Effort comparison



b) MMRE comparison



c) RMSE comparison

Fig. 1: Graphical representation of effort, MMRE and RMSE

Table 1. Effort estimated using various techniques

Project No.	Actual Size	Actual Effort	COCOMO	ACO	BCO	PSO
1	16.2	86.1	44.69	90.33	117.37	89.15
2	5.34	24.02	13.94	15.82	22.21	25.23
3	7.6	36.05	20.19	27.53	37.71	19.53
4	4.7	20.74	12.19	12.94	18.34	14.87
5	3.1	12.85	7.87	6.74	9.82	8.87
6	5.2	23.3	13.55	15.17	21.34	14.56
7	6.8	31.72	17.96	23.12	31.92	19.87
8	6.4	29.59	16.85	21.02	29.14	23.45
9	7.2	33.88	19.07	25.29	34.78	25.78
10	5.4	24.34	14.10	16.10	22.59	18.19
11	8.5	41.01	22.70	32.82	44.61	34.87
12	7.8	37.15	20.74	28.67	39.21	26.61
13	12.5	63.9	34.04	60.12	79.55	34.45
14	10.4	51.71	28.06	45.05	60.37	58.47
15	9.5	46.6	25.52	39.08	52.71	42.35
16	3.4	14.29	8.67	7.79	11.28	10.18
17	6.8	31.73	17.96	23.12	31.92	25.48
18	5.8	26.42	15.20	18.01	25.14	19.18
19	7.4	34.96	19.63	26.40	36.23	28.56
20	7.2	33.88	19.07	25.29	34.78	30.87
21	8.6	41.56	22.98	33.42	45.40	34.23
22	6.4	29.59	16.85	21.02	29.14	18.67
23	10.6	52.86	28.63	46.41	62.12	50.76
24	6.3	29.06	16.58	20.51	28.46	23.34
25	4.5	19.73	11.64	12.09	17.18	14.29
26	9.7	47.73	26.08	40.38	54.38	43.89
27	8.4	40.45	22.42	32.21	43.82	40.23
28	6.2	28.53	16.30	20.00	27.79	17.98
29	8.5	41.01	22.70	32.82	44.61	35.43
30	6.2	28.53	16.30	20.00	27.79	24.32
31	2.6	10.5	6.55	5.11	7.55	6.44
32	2.5	10.03	6.28	4.80	7.12	5.78
33	4.3	18.73	11.10	11.26	16.05	23.34
34	4.6	20.24	11.92	12.52	17.76	14.32
35	6.6	30.65	17.41	22.06	30.52	24.46
36	7.4	34.96	19.63	26.40	36.23	25.67
37	4.6	20.24	11.92	12.52	17.76	14.45
38	8.6	41.56	22.98	33.42	45.40	46.34
39	5.5	24.85	14.37	16.57	23.22	28.89
40	4.8	21.25	12.46	13.38	18.93	24.56
41	18	97.19	49.92	106.58	137.46	138.89
42	12.5	63.9	34.04	60.12	79.55	84.88
43	6.7	31.19	17.68	22.59	31.22	34.67
44	8.4	44.7	22.42	32.21	43.82	46.45
45	5.7	22.4	14.92	17.52	24.50	28.46
46	2.8	9.4	7.08	5.74	8.43	9.8
47	6.4	24	16.85	21.02	29.14	25.45
48	9.1	49.2	24.39	36.53	49.41	35.68

REFERENCES

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

[2] Azzeh, M.: A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation. Empirical Software Engineering 17(1-2), 90–127 (2012).

[3] Bhatnagar, R., Bhattacharjee, V., Ghose, M.K.: Software development effort estimation neural network vs. regression modeling approach. International Journal of Engineering Science and Technology 2(7), 2950–2956 (2010).

[4] Bhattacharya, P., Srivastava, P.R., Prasad, B.: Software Test Effort Estimation Using Particle Swarm Optimization. In: Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) held in Visakhapatnam, India, January 2012, pp. 827–835. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).

[5] Braga, P.L., Oliveira, A.L., Meira, S.R.: A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation. In: Proceedings of the 2008 ACM symposium on Applied computing, pp. 1788–1792 (2008).

[6] Braga, P.L., Oliveira, A.L.I., Ribeiro, G.H.T., Meira, S.R.L.: Bagging predictors for estimation of software project effort. In: International Joint Conference on Neural Networks, pp. 1595–1600. Florida,USA (2007).

[7] Burgess, C.J., Lefley, M.: Can genetic programming improve software effort estimation? A comparative evaluation. Information and Software Technology 43(14), 863 – 873 (2001).

[8] Chalotra, S., Sehra, S., Brar, Y., Kaur, N.: Tuning of COCOMO Model Parameters by using Bee Colony Optimization. Indian Journal of Science & Technology 8(14), 1–5 (2015).

[9] Chen, W.N., Zhang, J.: Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler. IEEE Transactions on Software Engineering 39(1), 1–17 (2013).

[10] Corazza, A., Martino, S.D., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E.: Using tabu search to configure support vector regression for effort estimation. Empirical Software Engineering 18(3), 506–546 (2013).

[11] Dewan, N., Sehra, S.K.: Ant Colony Optimization Based Software Effort Estimation. International Journal of Computer Science And Technology 5(3), 53–56 (2014).

[12] Dizaji, Z.A., Gharehchopogh, F.S.: A hybrid of ant colony optimization and chaos optimization algorithms approach for software cost estimation. Indian Journal of Science & Technology 8(2), 128–133 (2015).

[13] Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoretical Computer Science 344(2), 243 – 278 (2005).

[14] E. Mendes I. Watson, C.T.N.M.S.C.: A comparative study of cost estimation models for web hypermedia applications. Empirical Software Engineering 8(2), 163–196 (2003).

[15] Elish, M.: Improved estimation of software project effort using multiple additive regression trees. Expert Systems with Applications 36(7), 10,774–10,778 (2009).

[16] Ferrucci, F., Gravino, C., Sarro, F.: How multi-objective genetic programming is effective for software development effort estimation? In: Search Based Software Engineering, pp. 274–275. Springer (2011).

[17] Gupta, R., Chaudhary, N., Pal, S.K.: Hybrid model to improve bat algorithm performance. In: International Conference on Advances in Computing, Communications and Informatics, pp. 1967–1970. New Delhi, India (2014).

[18] Huang, S.J., Chiu, N.H., Chen, L.W.: Integration of the grey relational analysis with genetic algorithm for software effort estimation. European Journal of Operational Research 188(3), 898–909 (2008).

[19] Jørgensen, M.: Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation. Information and Software Technology 53(12), 1382–1390 (2011).

[20] Jørgensen, M.: A review of studies on expert estimation of software development effort. Journal of Systems and Software 70(1), 37–60 (2004).

[21] Jørgensen, M., Moløkken-Østvold, K.: How large are software cost overruns? A review of the 1994 {CHAOS} report. Information and Software Technology 48(4), 297 – 301 (2006).

[22] Kaur, M., Sehra, S.K.: Particle swarm optimization based effort estimation using Function Point analysis. In: Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on, pp. 140–145 (2014).
[23] Kennedy, J., Eberhat, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948 (1995).
[24] Menzies, T., Chen, Z., Hihn, J., Lum, K.: Selecting best practices for effort estimation. Software Engineering, IEEE Transactions on 32(11), 883–895 (2006).
[25] Oliveira, A.L.I., Braga, P.L., Lima, R.M.F., Corn´elio, M.L.: GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Information and Software Technology 52(11), 1155 – 1166 (2010).
[26] Park, H., Baek, S.: An empirical validation of a neural network model for software effort estimation. Expert Syst. Appl. 35(3), 929–937 (2008).
[27] Pertiwi, A.P., Suyanto: Globally Evolved Dynamic Bee Colony Optimization, pp.52–61. Springer Berlin Heidelberg, Berlin, Heidelberg (2011).
[28] Prasad Reddy P.V.G.D, Hari CH.V.M.K., Srinivasa Rao T.: Multi Objective Particle Swarm Optimization for Software Cost Estimation. International Journal of Computer Applications (0975 – 8887) 32(3), 13–17 (2011).
[29] Sarro, F.: Search-based Approaches for Software Development Effort Estimation. In: Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, Profes '11, pp. 38–43. ACM, New York, NY, USA (2011).
[30] Sheta, A., Rine, D., Ayesh, A.: Development of software effort and schedule estimation models using Soft Computing Techniques. In: Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, pp. 1283–1289 (2008).
[31] Sheta, A.F.: Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. Journal of Computer Science 2(2), 118–123 (2006).
[32] Srivastava, D.K., Chauhan, D.S., Singh, R.: VRS Model: A Model for Estimation of Efforts and Time Duration in Development of IVR Software System. Int. J. of Software Engineering, IJSE 5(1) (2012).