Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 167 (2020) 2643–2652

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2019)

Estimation of Software Development Effort: A Differential Evolution Approach

Prerna Singal[a,*], A Charan Kumari[b], Prabha Sharma[a]

[a]*The NorthCap University, Gurugram – 122001, India*
[b]*Dayalbagh Educational Institute, Agra, India*

Abstract

Several software effort estimation techniques based on mathematical formulations have been proposed, however there is little consensus on the best technique for effort estimation, which can predict the estimated effort optimally for any given project. Much research has been carried out in finding out the most suitable parameter values for accurate effort estimation. This research investigates the efficacy of Differential Evolution algorithms in improving the parameter values for algorithmic models like CoCoMo and CoCoMo II. Parameter values were obtained by using the three successful mutation strategies in Differential Evolution. The proposed methodology was tested on two datasets from the Promise Repository. Test results were compared with the original CoCoMo and CoCoMo II models based on MMRE. The proposed differential evolution approach predicted the estimated effort more accurately than the original models for the two datasets under consideration.

Keywords: CoCoMo; CoCoMo II; Differential Evolution; Algorithmic models

* Corresponding author. Tel.: 9953591373
E-mail address: prernasingal@ncuindia.edu

1877-0509
10.1016/j.procs.2020.03.343

1. Introduction

Software projects require efficient planning for a timely delivery, which is within budget and as per the desired quality standards. To achieve this, project managers need reliable techniques for conducting project feasibility studies, project planning, resource allocation and overall project cost estimation and scheduling. Substantial research has been done to improve the effort estimation accuracy for developing any software project.

Algorithmic models for software effort estimation were developed on the fact that effort required to develop a project will depend on factors like size of the project measured in delivered source lines of code, development environment, personnel involved in development, technology used, experience of the personnel and their expertise. These factors or cost drivers determine the development effort. Thus development effort could be estimated as a function of these cost drivers. Going forward with this notion, projects started collecting data related to the project planning and development, with the earliest reported analysis in 1956 by Benington [4], which linked the size of the project i.e. the delivered source lines of code with development productivity, support software and project documentation.

In the year 1964, U.S Air force came together with System Development Corporation to carry out a historical project in the field of software development effort estimation. The main aim of the project was to collect project data on various cost factors affecting the development effort. This data then could be utilized for studying and analyzing the impact of different cost drivers on the estimated effort. In 1965, SDC cost model by Nelson [27] was proposed, with 13 cost drivers, which established a linear relationship between predicted effort and the cost drivers. It created a database of around 169 projects but the model did not produce very accurate results. In 1970's, TRW Wolverton [39] related the estimated effort per project with relative degree of difficulty, project novelty and the type of the project. Then several other algorithmic models were proposed in the 1970's, finally leading to the constructive cost model by Boehmn [7] popularly known as the CoCoMo Model.

The motivation behind the CoCoMo model was to make people understand the economies of commissioning, developing and supporting any software project. It not only provides an estimate for the required development effort but it also provides the various cost factors which make up the cost of the project. CoCoMo proposed a nonlinear relationship between the size of the project and estimated effort. Size was measured in delivered source lines of code and estimated effort in man months (MM). With the advent of component based development and service oriented architecture, Boehmn enhanced the CoCoMo model to CoCoMo II to incorporate the changes around code reuse, function points and other related changes.

CoCoMo II was also formulated with a nonlinear relationship between the size of the project and estimated effort. The non-linearity was accounted to certain scale factors, which were calibrated based on the project data. This was a major shift from the original CoCoMo model, where non-linearity was attributed to the nature of the project. The scale factors were assigned weights with values of the form: very low, low, nominal, high, very high and extra high. Algorithmic models improved the estimation accuracy and also the availability of historical project data in the form of publically available data repositories like the Promise Repository. The researchers could utilize the data to further improve the estimation accuracy. This gave way to application of machine learning techniques in the field of software effort estimation.

Existing algorithmic models have been refined using the machine learning and evolutionary techniques like genetic algorithms [8], neural nets [14, 23, 30, 37], swarm optimization [29, 33, 38], support vector machines [9, 13], fuzzy logic [2, 18, 21, 34] and many more machine learning techniques. During the first international contest on evolutionary optimization in Japan, Differential Evolution was proposed by Storn and Price [35]. They proposed a differential operator instead of the conventional crossover and mutation operator in genetic algorithms. This led to low computational complexity, faster convergence and optimized memory utilization [20, 31, 35] as compared to other evolutionary approaches like genetic algorithms, simulated annealing or tabu search.

In this context, this research explores the effectiveness of DE in software effort estimation using algorithmic models (CoCoMo and CoCoMo II). Rest of the paper illustrates the related work in the field, introduction to the proposed approach, experimental setup, results, discussion and conclusion.

2. Related Work

Jorgensen [19] conducted a systematic review of 304 research papers published in over 76 journals in 2004 and broadly classified the software effort estimation techniques in two categories: parametric models and machine learning models. Parametric Models were based on some mathematical formula or algorithm, where estimated effort is a function of certain parameters and variables. Machine learning models were kind of black box optimization models using techniques like artificial intelligence (AI) and artificial neural networks (ANN). Evolutionary algorithms were used to optimize the parameter values [10, 11, 21, 25] in the algorithmic models.

Oliviera [28] proposed a genetic algorithm (GA) based approach to tune the parameter values of the regression model for estimating effort. They further extended the research by feature subset selection using GA i.e. to find a subset of the parameters which mainly impact the estimated effort. Sultan [1] explored Differential evolution to improve the parameter values for CoCoMo Intermediate model for an industrial dataset. In his research, he compared the results obtained by applying DE and particle swarm optimization (PSO) on CoCoMo, Doty, SLIM and various other models. Nasar [26] investigated the usage of DE in optimizing the software testing process. The main objective of his research was the optimal distribution of testing load amongst all the available personnel.

Thamarai [36] applied DE for optimizing the similarity measure in analogy based effort (ABE) estimation. The results were investigated on the Desharnais dataset (Promise Repository) using the cost functions as MMRE and PRED (25). He further compared the results with the firefly and hybrid fuzzy logic algorithms. The algorithms improved the estimation accuracy for ABE on the Desharnais dataset. Benala [3] published an extensive research in 2018 comparing the effects of DE on effort estimation with that of hybrid PSO, ANN algorithms on the Promise Repository datasets. In his research, Benala experimented with the five mutation strategies available in DE. This research aims to improve the parameter values of the CoCoMo and CoCoMo II models using a differential evolution algorithm based approach. Our test results show that the accuracy of the proposed algorithm goes up.

3. Background

3.1. Algorithmic Models for Software Development Effort Estimation

These models expresses the software development effort estimate as a function of the variables on which the development effort depends. These variables are termed as cost drivers which impact the software development effort estimation. Strength of an algorithmic model lies in its objective, repeatability and analyzability [7]. Such models have proved to be efficient and good for sensitivity analysis. Algorithmic models like Constructive Cost Model (CoCoMo) have been calibrated using the collected project data which is now available in the Promise Repository. These models have their own shortcomings due to being calibrated to past data not the future data, subjective inputs and lack of assessment of exceptional circumstances. In the past, numerous algorithmic models have been suggested namely the Putnam SLIM Model [32], Doty Model [17], RCS Price S Model [12], Constructive cost Model (CoCoMo) [7] and Constructive Cost Model II (CoCoMo II) [5]. This research focuses on CoCoMo and CoCoMo II Model for application of Differential Evolution.

3.2. CoCoMo

CoCoMo enables to ascertain why the cost has been incurred and what its impact on the required development effort is. CoCoMo estimates the development effort as a function of software size (measured in KDSI) multiplied by the software cost factors. The model was first proposed by Boehm [6, 7] which was calibrated using the data from around 63 projects being developed by NASA. Estimated development effort was calculated based on the

development mode of the project – Organic, Semi-Detached or embedded. CoCoMo Model established a non-linear relationship between the size of the project and the estimated effort:

$$(MM)_{Nom} = a*(KDSI)^{b} \ , \quad \text{---------- (1)}$$

where Estimated effort was expressed in Man Months (MM), {a, b} are the constants whose values depend upon the development mode of the project. KDSI refers to the size of the project measured in thousands of delivered source instructions. Table 1 lists the values of a, b for organic, semi-detached and embedded projects.

Table 1. CoCoMo Nominal Effort Equations

Development Mode	Estimated Effort
Organic	$(MM)_{Nom} = 3.2*(KDSI)^{1.05}$
Semidetached	$(MM)_{Nom} = 3.0*(KDSI)^{1.12}$
Embedded	$(MM)_{Nom} = 2.8*(KDSI)^{1.20}$

CoCoMo Model identified 15 cost drivers which could impact the estimated effort for any software project. These cost drivers were assigned weights which could be multiplied by the estimated effort based on the cost driver rating (very low, low, nominal, high, very high, extra high). The model also calibrated the values for the 15 cost drivers which are depicted in Table 2.

Table 2: CoCoMo Cost Drivers [7]

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product Attributes						
RELY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0,85	1.00	1.15	1.30	1.65
Computer Attributes						
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
Personnel Attributes						
ACAP	1.46	1.19	1.00	0.86	0.71	
AEXP	1.29	1.13	1.00	0.91	0.82	
PCAP	1.42	1.17	1.00	0.86	0.70	
VEXP	1.21	1.10	1.00	0.90		
LEXP	1.14	1.07	1.00	0.95		
Project Attributes						
MODP	1.24	1.10	1.00	0.91	0.82	
TOOL	1.24	1.10	1.00	0.91	0.83	
SCED	1.23	1.08	1.00	1.04	1.10	

3.3. CoCoMo II

With the advent of parallel rapid development models, reusable components, reengineering and commercially off the shelf (COTS) software, there was a need to extend the existing CoCoMo model to include the effect of these new methods on software effort estimation. Depending upon the phase of the software development, CoCoMo II can be defined for Early Design or Post-architectural model [5]. Early Design model is used for high level effort

estimates wherein various architectural and design approaches can be explored with respect to the estimated development effort. On the other hand, Post-architectural model comes in handy when the architecture of the application has been decided and a more refined effort estimate is needed. Both Early Design and Post-Architecture model utilize the same mathematical formula to estimate the development effort.

$$PM_{NS} = A * Size^{E} * \prod_{i=1}^{n} EM_i \quad \text{----------(2)}$$

$$\text{where } E = B + 0.01 * \sum_{j=1}^{5} SF_j \quad \text{----------(3)}$$

In the above mathematical formula, n represents the number of effort multipliers considered for effort estimation: n is 16 for Post-architectural model and 6 for Early Design. SF stands for the cost drivers which impact the estimated effort exponentially also known as scale factors. There are five scale factors in CoCoMo II Model namely precedentedness (PREC), development flexibility (FLEX), Risk Resolution (RESL), Team Cohesion (TEAM) and Process Maturity (PMAT). In the equations, A and B are constants whose values have been calibrated using data from 161 projects. These values can also be calibrated according to the local project environment. The values of the effort multipliers have been calibrated using the data from 161 projects in the CoCoMo research. Table 3 lists the values of the effort multipliers used in CoCoMo II Model.

Table 3: CoCoMo II Effort Multipliers [5]

Scale Factors	Very Low	Low	Nominal	High	Very High	Extra High
RELY	0.82	0.92	1.00	1.10	1.26	
DATA		0.90	1.00	1.14	1.28	
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE		0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	
TIME			1.00	1.11	1.29	1.63
STOR			1.00	1.05	1.17	1.46
PVOL		0.87	1.00	1.15	1.30	
ACAP	1.42	1.19	1.00	0.85	0.71	
PCAP	1.34	1.15	1.00	0.88	0.76	
PCON	1.29	1.12	1.00	0.90	0.81	
APEX	1.22	1.10	1.00	0.88	0.81	
PLEX	1.19	1.09	1.00	0.91	0.85	
LTEX	1.20	1.09	1.00	0.91	0.84	
TOOL	1.17	1.09	1.00	0.90	0.78	
SITE	1.22	1.09	1.00	0.93	0.86	0.80
SCED	1.43	1.14	1.00	1.00	1.00	

3.4. Differential Evolution

Differential Evolution (DE) is a heuristic approach used for minimizing non-linear and non-differentiable continuous functions. This approach is utilized for finding the global optima for a given problem, i.e. optimizing certain system properties based on the given parameters specific to the system. DE Algorithm first initializes the initial population in the form of trail vectors represented by $x_{i,G}$ covering all the parameter space. As a thumb rule, all the random choices for the initial population follow a uniform probability distribution, i.e. any vector is equally likely to be chosen. Then, a mutated vector is generated by adding the weighted difference of two randomly chosen vectors to a third parameter vector. This mutant vector is termed as $v_{i,G+1}$, where G+1 signifies a vector for the next generation. Then a crossover operation on the mutant vectors is carried out and termed as a trial vector represented

by $u_{i,G+1}$. Finally, selection is done between the trial and the target vector. This evolution process is repeated until the desired fitness function vale is achieved, or the pre-defined number of iterations for the population generation have been exhausted.

Table 4: Differential Evolution Algorithm Steps

Step 1: Initialization	
	Initialize the population weight vectors $x_{i,G}$ Where i = 1, 2, 3,…., NP
Step 2: Mutation	
	Generate a Mutant vector $v_{i,G+1}$ for each target vector $v_{i,G+1} = x_{r1,G} + F.(x_{r2.G} - x_{r3.G})$ Where r1, r2, r3 ϵ {1,2,…..,NP}, mutually different, F > 0
Step 3: Crossover	
	Generate a Trial vector $u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, ..., u_{Di,G+1})$ Where $u_{ji,G+1} = \begin{cases} v_{ji,G+1} \ if \ (randb\ (j) \le \text{CR}) \text{ or } j = rnbr(i) \\ u_{ji,G} \ if \ (randb\ (j) > \text{CR}) \text{ or } j \ne rnbr(i) \end{cases}$ j= 1,2,….,D
Step 4: Selection	
	Compare trial vector $u_{i,G+1}$ with the target vector $x_{i,G}$ using Greedy Criteria Select the value with smaller cost function
Step 5: Iteration	
	Repeat steps 1-4, until desired fitness function value reached or pre-defined number of Generations have been generated

This algorithm given by Rainer Storm and Kenneth Price [35] is known to converge fast and give better results among other evolutionary approaches. Like other evolutionary approaches, DE is also a population based algorithm which evolves a population of NP dimensional vectors popularly called chromosomes. These chromosomes hold the key information and candidate solutions represented as $x_{i,G}$ where i = 1, 2, 3,…., NP. Table 4 demonstrates the steps in DE Algorithm in the form of a flowchart. Rainer and Storm compared DE with existing evolutionary approaches and simulated annealing, and found that DE converges faster and gives global optima with right set of control parameters.

4. Experimental Setup

Differential Evolution was applied to CoCoMo and CoCoMo II Models respectively using the datasets from the Promise Repository. Two datasets were selected (cocomo81, nasa93) as they had the required cost driver variables to be optimized as part of the research. Mean magnitude of relative error (MMRE) was used as the fitness function with an objective to minimize the cost of the function value. After extensive experimentation, DE parameters were set at: F (DE Step size) = 0.3, CR (Crossover probability constant) = 1, N (Population) = 100, G (Iterations or generation) = 50. Differential Evolution [3] was applied using three mutation strategies known for converging to global optima faster.

Table 5: DE Mutation Strategies

Notation	Mutation Strategy	Mathematical Formula
DE 1	DE/rand/1/bin	$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}), r1 \ne r2 \ne r3 \ne i$

DE 2	DE/rand/2/bin	$v_{i,G+1} = x_{r1,G} + F.(x_{r2,G} - x_{r3,G}) + F.(x_{r4,G} - x_{r5,G})$ $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$
DE 3	DE/best/1/bin	$v_{i,G+1} = x_{best,G} + F.(x_{r1,G} - x_{r2,G}), r1 \neq r2 \neq i$

For CoCoMo, initial population vector for the experiment is constructed from the cost driver values mentioned in Table 2. Similarly, for CoCoMo II, initial population vector is constructed using the cost driver values picked from Table 3. Fitness function for the setup is described in section 4.2, with the objective of minimizing the MMRE in predicting the estimated effort for software development projects.

4.1. Datasets

Two datasets Cocomo81 and Nasa93 from the (http://Promise.site.uottawa.ca/SERepository/) Promise [22] Repository, were chosen for the research to validate the model. Both the datasets include projects developed in the United States. Cocomo81 has 63 and Nasa93 has data for 93 projects respectively. Cocomo81 dataset has 17 features while the Nasa93 dataset has 24 features. The datasets in the Promise Repository have a skewness [24] factor of around 6.6, so they are not normally distributed. This results in inaccuracies in predicting software development effort.

4.2. Evaluation Criteria

This research utilizes the Mean Magnitude of relative error (MMRE) as the fitness function to evaluate the accuracy of estimated effort. MMRE is defined as the mean of the difference between the estimated and actual effort values for all the projects given by the mathematical formula. In the formula N represents the total number of projects.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}(y_{i,actual} - y_{i,estimated}) / y_{i,actual} \quad \text{.....(4)}$$

where $y_{i,actual}$ represents the actual effort spent on the project and $y_{i,estimated}$ represents the estimated effort for the project using either CoCoMo equation (1) or CoCoMo II equation (2, 3).

4.3. Results

This section presents the results of applying DE on CoCoMo and CoCoMo II Models respectively. All the three mutation strategies were applied on the models. Population size was set at 100 and number of iterations for each run was set to 50. First the approach was tested on cocomo81 dataset, once utilizing the CoCoMo Model and then CoCoMo II Model. Similarly proposed approach was tested on nasa93 dataset, once calculating the estimated effort with CoCoMo Model and then with CoCoMo II Model. Then the results were compared with the estimated effort obtained using the original CoCoMo & CoCoMo II model. Table 6 illustrates the results obtained.

Table 6: DE Applied on CoCoMo and CoCoMo II Models

Model	***Dataset***	***Mutation Strategy***	***MMRE (proposed DE based approach)***	***MMRE (original)***
CoCoMo	**Cocomo81**	DE 1	0.37661	0.408
		DE II	0.34749	0.408
		DE III	**0.29885**	0.408

CoCoMo II	**Nasa 93**	DE 1	0.47812	0.456
		DE II	0.37493	0.456
		DE III	0.35379	0.456
	Cocomo81	DE 1	0.58376	0.498
		DE II	0.49378	0.498
		DE III	0.30927	0.498
	Nasa 93	DE 1	0.33867	0.445
		DE II	0.31846	0.445
		DE III	**0.22867**	0.445

On the cocomo81 dataset, CoCoMo model showed better results than the CoCoMo II Model with respect to MMRE. For mutation strategy DE3 (DE best) MMRE obtained was 0.29885. It should be noted that cocomo81 dataset was developed using the CoCoMo cost drivers only. On the other hand, for nasa93 dataset, CoCoMo II showed better results. For mutation strategy DE3, MMRE obtained was 0.22867 which is a bit lower than the MMRE obtained for CoCoMo model (0.33867). Again, worth noting nasa93 dataset had cost driver values calibrated to CoCoMo II Model. Figure 1 illustrates a graphical comparison of the experimental results obtained.

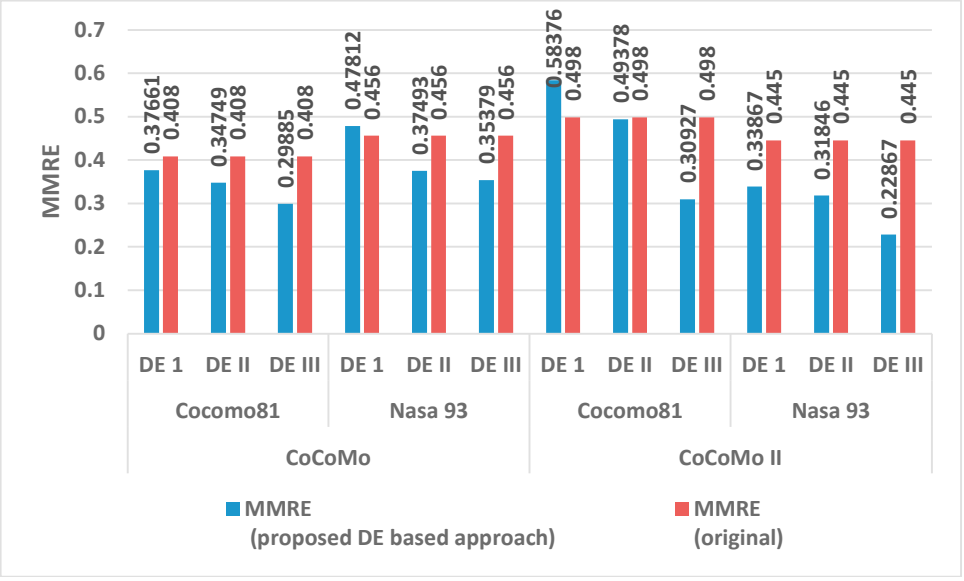

Figure 1: MMRE comparison for CoCoMo and CoCoMo II Models for Cocomo81 and Nasa93 datasets

5. Discussion

This research has focused on the application of DE on the algorithmic models such as CoCoMo and CoCoMo II. The proposed approach was tested rigorously on the Promise Repository datasets. Results indicate that DE III (DE/best/1/bin) gives the best results for the estimated effort. Our experimental results indicate that the performance of DE on algorithmic models for effort estimation is satisfactory. The DE algorithm comes with some inherent benefits [31] like faster convergence rate, low computation complexity, optimal memory utilization and low computation effort. The research could be further extended to improve the estimation accuracy, and at the same time improving the productivity of the project using DE. Also, the cost driver weight values could be optimized using the DE for improved accuracy in software effort estimation. In this research, MMRE was used as the fitness function, other measures of accuracy could be used for improving the accuracy of the predicted effort.

6. Conclusion

Based on the experimental results carried out on the Cocomo81 & Nasa93 datasets, it can be inferred that DE based effort estimation approach provides a better effort estimate as compared to the original CoCoMo and CoCoMo II models. DE algorithm provides improved cost driver values for both the models, thus improving the effort estimation accuracy.

References

[1] S. Aljahdali, "Development of a software effort estimation model using differential evolution," Journal of Electronics and Computer Science, vol. 12, pp. 1-8, 2010.

[2] M. Azzeh, D. Neagu, and P. I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers," Journal of Systems and Software, vol. 84, pp. 270-284, 2011.

[3] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," Swarm and Evolutionary Computation, vol. 38, pp. 158-172, 2018.

[4] H. D. Benington, "Production of large computer programs," Annals of the History of Computing, vol. 5, pp. 350-361, 1983.

[5] B. Boehm, C. Abts, B. Clark, and S. Devnani-Chulani, "COCOMO II model definition manual," The University of Southern California, 1997.

[6] B. W. Boehm, Software engineering economics vol. 197: Prentice-hall Englewood Cliffs (NJ), 1981.

[7] B. W. Boehm, "Software engineering economics," IEEE transactions on Software Engineering, pp. 4-21, 1984.

[8] C. J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation," Information and software technology, vol. 43, pp. 863-873, 2001.

[9] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, "Using tabu search to configure support vector regression for effort estimation," Empirical Software Engineering, vol. 18, pp. 506-546, 2013.

[10] S. Dalal, N. Dahiya, and V. Jaglan, "Efficient Tuning of COCOMO Model Cost Drivers Through Generalized Reduced Gradient (GRG) Nonlinear Optimization with Best-Fit Analysis," in Progress in Advanced Computing and Intelligent Engineering, ed: Springer, 2018, pp. 347-354.

[11] L. P. dos Santos and M. G. V. Ferreira, "Safety Critical Software Effort Estimation using COCOMO II: A Case Study in Aeronautical Industry," IEEE Latin America Transactions, vol. 16, pp. 2069-2078, 2018.

[12] F. R. Freiman and R. Park, "Price software model-version 3: An overview," in Proceedings, ieee/piny workshop on quantitative software models, ieee catalog no. th0067, 1979, pp. 32-44.

[13] A. García-Floriano, C. López-Martín, C. Yáñez-Márquez, and A. Abran, "Support vector regression for predicting software enhancement effort," Information and Software Technology, vol. 97, pp. 99-109, 2018.

[14] A. Garcia, I. Gonzalez, R. Colomo-Palacios, J. L. Lopez, and B. Ruiz, "Methodology for software development estimation optimization based on neural networks," IEEE Latin America Transactions, vol. 9, pp. 384-398, 2011.

[15] W. Gong, Z. Cai, C. X. Ling, and H. Li, "Enhanced differential evolution with adaptive strategies for numerical optimization," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 41, pp. 397-413, 2010.

[16] S.-M. Guo, C.-C. Yang, P.-H. Hsu, and J. S.-H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," IEEE Transactions on Evolutionary Computation, vol. 19, pp. 717-730, 2014.

[17] P. G. Hamer and G. D. Frewin, "MH Halstead's Software Science-a critical examination," in Proceedings of the 6th international conference on Software engineering, 1982, pp. 197-206.

[18] X. Huang, D. Ho, J. Ren, and L. F. Capretz, "Improving the COCOMO model using a neuro-fuzzy approach," Applied Soft Computing, vol. 7, pp. 29-40, 2007.

[19] M. Jørgensen, "Top-down and bottom-up expert estimation of software development effort," Information and Software Technology, vol. 46, pp. 3-16, 2004.

[20] D. Karaboğa and S. Ökdem, "A simple and global optimization algorithm for engineering problems: differential evolution algorithm," Turkish Journal of Electrical Engineering & Computer Sciences, vol. 12, pp. 53-60, 2004.

[21] M. Kazemifard, A. Zaeri, N. Ghasem-Aghaee, M. A. Nematbakhsh, and F. Mardukhi, "Fuzzy emotional COCOMO II software cost estimation (FECSCE) using multi-agent systems," Applied Soft Computing, vol. 11, pp. 2260-2270, 2011.

[22] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," Journal of Systems and Software, vol. 86, pp. 1879-1890, 2013.

[23] C. López-Martín and A. Abran, "Neural networks for predicting the duration of new software projects," Journal of Systems and Software, vol. 101, pp. 127-135, 2015.

[24] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The Promise Repository of empirical software engineering data," ed: June, 2012.

[25] D. Nandal and O. P. Sangwan, "Software Cost Estimation by Optimizing COCOMO Model Using Hybrid BATGSA Algorithm," International Journal of Intelligent Engineering and Systems, vol. 11, pp. 250-263, 2018.

[26] M. Nasar, P. Johri, and U. Chanda, "A differential evolution approach for software testing effort allocation," Journal of Industrial and Intelligent Information Vol, vol. 1, 2013.

[27] E. A. Nelson, "Management handbook for the estimation of computer programming costs," SYSTEM DEVELOPMENT CORP SANTA MONICA CA1967.

[28] A. L. Oliveira, P. L. Braga, R. M. Lima, and M. L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," information and Software Technology, vol. 52, pp. 1155-1166, 2010.

[29] C. Pareta, N. Yaadav, A. Kumar, and A. K. Sharma, "Predicting the Accuracy of Machine Learning Algorithms for Software Cost Estimation," in Emerging Trends in Expert Applications and Security, ed: Springer, 2019, pp. 605-615.

[30] H. Park and S. Baek, "An empirical validation of a neural network model for software effort estimation," Expert Systems with Applications, vol. 35, pp. 929-937, 2008.

[31] K. Price, R. M. Storn, and J. A. Lampinen, Differential evolution: a practical approach to global optimization: Springer Science & Business Media, 2006.

[32] L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE transactions on Software Engineering, pp. 345-361, 1978.

[33] D. L. Shrestha and D. P. Solomatine, "Machine learning approaches for estimation of prediction interval for the model output," Neural Networks, vol. 19, pp. 225-235, 2006.

[34] P. R. Sree and R. SNSVSC, "Improving efficiency of fuzzy models for effort estimation by cascading & clustering techniques," Procedia Computer Science, vol. 85, pp. 278-285, 2016.

[35] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," Journal of global optimization, vol. 11, pp. 341-359, 1997.

[36] I. Thamarai and S. Murugavalli, "An Evolutionary Computation Approach for Project Selection in Analogy based Software Effort Estimation‖," Indian Journal of Science and Technology, vol. 9, 2016.

[37] Z. H. Wani and S. Quadri, "Software Cost Estimation Based on the Hybrid Model of Input Selection Procedure and Artificial Neural Network," Artificial Intelligent Systems and Machine Learning, vol. 10, pp. 18-24, 2018.

[38] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," Information and Software Technology, vol. 54, pp. 41-59, 2012.

[39] R. W. Wolverton, "The