Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 5 (1): 38-48, 2011
ISSN 1819-4311 / DOI: 10.3923/jse.2011.38.48

A Fine Parameter Tuning for COCOMO 81 Software Effort Estimation using Particle Swarm Optimization

[1]CH.V.M.K. Hari and [2]P.V.G.D. Prasad Reddy
[1]Department of Information Technology, Gitam University, Visakhapatnam, India
[2]Department of Computer Science and Software Engineering, Andhra University, Visakhapatnam, India

Corresponding Author: CH.V.M.K. Hari, Department of Information Technology, Gitam University, Visakhapatnam, India

ABSTRACT

The basic goal of project planning is to look into the future, identify the activities that need to be done to complete the project successfully and plan scheduling and resource allocation for these activities. Software effort estimation is the most important activity in project planning. So far many models are proposed by using machine learning algorithms, but no model is proved successful for efficiently and consistently predicting the effort. In this study we proposed two models using particle swarm optimization (PSO) with Constriction Factor for fine tuning of parameters of the constructive cost model (COCOMO) effort estimation. The models deals efficiently with imprecise and uncertain input and enhances the reliability of software effort estimation. The experimental part of the study illustrates the approach and contrast it with the standard numeric version of the COCOMO, standard singal variable models, Triangular Membership Function and Gbell function Models.

Key words: Person months, kilo delivered lines of code, constructive cost model, particle swarm, optimization, software effort estimation

INTRODUCTION

Software project management activities are mainly classified into three categories: project planning, project monitoring and control and project termination. In project planning, cost estimation is one of the most important activities. Software cost estimation is the process of predicting how much amount of effort is required to build software. The effort is measured in terms of Person-Months (PM) thus later on it is converted into actual dollar-cost. The basic input parameters for software cost estimation is size, measured in KDLOC (Kilo Delivered Lines of Code). A number of models have been evolved to establish the relation between Size and Effort (Poli *et al*., 2007). The parameters of the algorithms are tuned using Regression Analysis (Sengupta, 2010) Genetic Algorithms (Sheta, 2006), Fuzzy models (Huang *et al*., 2006), Soft-Computing Techniques (Heng *et al*., 2006; Kaur and Singh, 2010), Computational Intelligence Techniques, Heuristic Algorithms, Neural Networks, Radial Basis and Regression (Idri *et al*., 2006; Jorgensen and Shepperd, 2007).

COCOMO effort estimation: Boehm (1981) derived a cost model called COCOMO (Constructive Cost Estimation Model) using data from a large set of projects at Teen Red Week TRW Software Production System (SPS) consulting firm based in California. The models are classified into:

- Original COCOMO Model

- COCOMO-II Model

Original COCOMO: It is a collection of three models. A basic model that is applied early in the project. An intermediate model which is applied after requirements. Advanced model is applied after design is complete.
The cost model is:

$$E = a^* (S)^b * EAF \quad (1)$$

Where:
S = Size is measured in KDLOC
E = Effort in terms of person months
EAF = Effort adjustment factor (Equal to 1 for basic model)

The factors a and b depend on the development mode. Boehm (1981) has defined three development modes (Bailey and Basili, 1981)

- Organic mode (for simple projects)
- Semi-detached mode(for intermediate projects)
- Embedded mode (for tight set of requirements, large projects)

Intermediate COCOMO: The intermediate COCOMO model computes effort as a function of program size and set of cost drivers. The intermediate COCOMO equation is:

$$E = a^* (S)^b * EAF$$

The parameters a, b are measured by using regression analysis and shown in Table 1, where a is the amplitude and b is the exponent.

The Effort Adjustment Factor (EAF) is calculated using 15 cost drivers (Poli *et al*., 2007). The cost drivers are grouped into four categories: Product, Computer, Personal and Project. Each cost driver is rated on a six-point ordinal scale ranging from low to high importance. Based on the rating an effort multiple is determined using the Table 2. The product of all effort multipliers is the EAF.

PSO with constriction factor: Another PSO (Iraj and Saleh, 2008; Sutha and Kamaraj, 2008), implementation dubbed PSO constriction coefficient was developed by Clerc (1999). Clerc modeled the particle swarm interactions using a set of complicated linear equations. Using a constriction coefficient results in particle convergence over time. That is amplitude of particle oscillations decreases as it focuses on the local neighborhood previous best points. Through the particle convergence to a point over time, the constriction coefficient also prevents a collapse if the right social conditions are in place. The particle will oscillate around the weighted mean of P_{id} and P_{gd},

Table 1: Modes of projects

Mode	a	b
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Table 2: Effort multipliers

		Rating				
Cost factors	Description	Very low	Low	Nominal	High	Very high
Product						
Rely	Required software reliability	0.75	0.88	1	1.15	1.40
Data	Database size	-	0.94	1	1.08	1.16
Cplx	Product complexity	0.70	0.85	1	1.15	1.30
Computer						
Time	Execution time constraint	-	-	1	1.11	1.30
Stor	Main storage constraint	-	-	1	1.06	1.21
Virt	Virtual machine volatility	-	0.87	1	1.15	1.30
Turn	Computer turnaround time	-	0.87	1	1.07	1.15
Personnel						
Acap	Analyst capability	1.46	1.19	1	0.86	0.71
Aexp	Application experience	1.29	1.13	1	0.91	0.82
Pcap	Programmer capability	1.42	1.17	1	0.86	0.70
Vexp	Virtual machine volatility	1.21	1.10	1	0.90	-
Lexp	Language experience	1.14	1.07	1	0.95	-
Project						
Modp	Modern programming practice	1.24	1.10	1	0.91	0.82
Tool	Software tools	1.24	1.10	1	0.91	0.83
Sced	Development schedule	1.23	1.08	1	1.04	1.10

if the previous best position and the neighborhood best position are far apart from each other the particle will perform an exploratory search (global search). During the search the neighborhood best position and previous best position will change and the particle will shift from local search back to global search. The constriction coefficient method therefore balances the need for local and global search depending on what social conditions are in place:

$$X_{id}(t) = f(\lambda, X_{id}(t-1), V_{id}(t-1), P_{id}, P_{gd})$$

$$\lambda = \frac{2k}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}$$

where:

$$\phi = c1 + c2,\ \phi > 4$$

$$V_{id}(t) = \lambda[\, V_{id}(t-1) + c_1\phi_1 (P_{id} - X_{id}(t-1)) + c_2\phi_2 (P_{gd} - X_{id}(t-1))] \quad (2)$$

$$X_{id}(t) = X_{id}(t-1) + V_{id} \quad (3)$$

Clerc (1999) found that by modifying ϕ, the convergence characteristics of the system can be controlled. Typically k = 1 and c1 =c 2 = 2 and ϕ is set to 4.1, then k = 0.73.

A FINE PARAMETER TUNING FOR COCOMO 81 SOFTWARE EFFORT ESTIMATION USING PSO

A typical cost model is derived using regression analysis on data collected from past software projects. In this model we tuned the parameters by using PSO with constriction factor. Effort is

plotted against the primary cost factor for a series of projects. The best fit of particles is then calculated among the data points. If the primary cost factor were a perfect prediction of effort, then every particle would be on the same place with derivative of velocity. The parameters are tuned very fast by considering the constriction factor.

Model description: In this model we consider PSO with constriction factor for faster convergence of COCOMO parameters.

Methodology (Algorithm)
Input: Size of Software Projects, Measured Efforts, Effort Adjustment Factor
Output: Optimized Parameters for Estimating Effort

The following is the methodology used to tune the parameters in the proposed models for Software Effort Estimation:

Step 1: Initialize swarm coefficients and other parameters k = 1 and c1 = c2 = 2, ϕ = 4.1 and k = 0.73
Step 2: Initialize particles velocity and position vectors randomly. Initialize these positions and velocity are personally best (P_{lb})
Step 3: for i=1 to n do

Evaluate the fitness function for each particle. We consider MARE is the fitness function.

Step 4: Find the locally best position of each particle by comparing the fitness value with the old value. New value is promising, then update the locally best position (P_{lb})
Step 5: Find the globally best particle among all locally best particles called globally best (P_{gb})
Step 6: In order to make faster convergence we consider the constriction factor which is evaluated with the following formula:

$$\lambda = \frac{2k}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}$$

Where:

$$\phi = c1 + c2,\ \phi > 4 \tag{4}$$

The particles velocity and positions are updated using $P_{lb,}$ P_{gb} with the following formula:

$$V_{id}(t) = \lambda[\, V_{id}(t\text{-}1) + c_1\phi_1 (P_{id}\text{-}X_{id}(t\text{-}1)) + c_2\phi_2 (P_{gd}\text{-}X_{id}(t\text{-}1))] \tag{5}$$

$$X_{id}(t) = X_{id}(t\text{-}1)\, V_{id}$$

where, I = a,b,c. (6)

Step 7: Step 4 to 6 are repeated until:

$$(W-\frac{1}{\lambda}) - \lambda c_1 r_1 - \lambda c_2 r_2 < 0$$

If the value is < 0 satisfied then PSO with constriction model is convergent or otherwise after some number of iterations.

Step 8: Giving Gbest solution parameters as optimal solution.
Step 9: Exit

PROPOSED MODELS

Model 1: According to COCOMO the cost estimation is a function of two variables, one is size another one EAF (Cost Multipliers):

$$E = a^* (S)^b * EAF$$

Now the parameters are tuned by using the above methodology.

Model 2: In order to reduce the uncertainty non linearity is existing at input level we use Biasing Effect.

So the equation is:

$$E = a^* (S)^b * EAF + c \tag{7}$$

MODEL ANALYSIS

The velocity and position update are as follows:

$$V_{id}(t) = \lambda [V_{id}(t-1) + c_1 \phi_1 (P_{id} - X_{id}(t-1)) + c_2 \phi_2 (P_{gd} - X_{id}(t-1))]$$

$$X_{id}(t) = X_{id}(t-1) + V_{id}$$

By considering the differential meta model (Zeng and Cui, 2007) for particle swarm optimization. The differential equations are:

$$\frac{d}{dt} V_{id}(t) = \lambda [(W-\frac{1}{\lambda}) V_{id}(t) + c_1 r_1 (P_{lb} - X_{id}(t)) + c_2 r_2 (P_{gb} - X_{id}(t))] \tag{8}$$

$$\frac{d}{dt} X_{id}(t) = V_{id}(t+1) \tag{9}$$

where, W =1 and:

$$\lambda = \frac{2k}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}}$$

$$\phi_{12} = c_1 r_1 + c_2 r_2$$

$$\frac{d}{dt}V_{id}(t) = [(1-\frac{1}{\lambda})V_{id}(t) + c_1r_1(P_{Ib}-X_{id}(t)) + c_2r_2(P_{gb}-X_{id}(t))]$$

$$= \lambda V_{id}(t)-V_{id}(t) + c_1 r_1 (P_{IB}-X_{id}(t)) + c_2 r_2 (P_{gd}-X_{id}(t))$$

$$\frac{d}{dt}V_{id}(t+1) = V_{id}(t) + \frac{d}{dt}V_{id}(t)$$

$$V_{id}(t+1) = \lambda [V_{id}(t) + \lambda c_1 r_1 (P_{Ib}-X_{id}(t)) + c_2 r_2 (P_{gd}-X_{id}(t))] \quad (10)$$

The standard state system of linear equation obtained:

$$\overline{Y(t)} = A.Y(t) + B.\mu(t)$$

Where:

$$Y(t) = \begin{bmatrix} V_{id}(t) \\ X_{id}(t) \end{bmatrix}$$

And:

$$\mu(t) = \begin{bmatrix} P_{Ib} \\ P_{gb} \end{bmatrix}$$

$$A = \begin{bmatrix} W(1-\frac{1}{\lambda}) & -\lambda(c_1r_1 + c_2r_2) \\ 1+W(1-\frac{1}{\lambda}) & -\lambda(c_1r_1 + c_2r_2) \end{bmatrix}$$

$$B = \begin{bmatrix} \lambda c_1 r_1 & \lambda c_2 r_2 \\ \lambda c_1 r_1 & \lambda c_2 r_2 \end{bmatrix}$$

And the equations are solved as follows:

$$Y(t) = e^{A(t-t_0)}Y(t_0) + \int_0^t e^{A(t-T)}.B.\mu(t).dt$$

From linear system theory if all Eigen values of matrix A have negative real part, the above equation is convergence. The Eigen equation of matrix A listed as follows:

$$\det(\lambda I - A) = 0$$

$$\det\begin{bmatrix} W(1-\frac{1}{\lambda})-\lambda & -\lambda(c_1r_1 + c_2r_2) \\ 1+W(1-\frac{1}{\lambda}) & -\lambda(c_1r_1 + c_2r_2)-\lambda \end{bmatrix}$$

$$\lambda^2 + \lambda(\lambda c_1 r_1 + \lambda c_2 r_2 - W(1 - \frac{1}{\lambda})) + (\lambda c_1 r_1 + \lambda c_2 r_2) = 0$$

The roots are:

$$\frac{-(\lambda c_1 r_1 + \lambda c_2 r_2 - W(1 - \frac{1}{\lambda})) \pm \sqrt{\left[\lambda c_1 r_1 + \lambda c_2 r_2 - W(1 - \frac{1}{\lambda})\right]^2 - 4(\lambda c_1 r_1 + \lambda c_2 r_2).1}}{2}$$

Therefore both Eigen values of matrix A have negative real part if:

$$(W - \frac{1}{\lambda}) - \lambda c_1 r_1 - \lambda c_2 r_2 < 0$$

is true.

In other words if:

$$(W - \frac{1}{\lambda}) - \lambda c_1 r_1 - \lambda c_2 r_2 < 0$$

is satisfied, it is convergent.

Finally globally best particle values are optimal tuned parameters. These parameters are used to estimate the effort of project. We have developed the above methodology for tuning parameters a, b and c in "C" Language. The performance measures we consider are MARE, VARE and VAF.

MODEL EXPERIMENTATION

For the study of these models we have taken dataset from COCOMO81 (Prasad Reddy, 2010) of 20 projects from NASA software project data are shown in Table 3. The tuning parameters for the PSO with constriction factor evolutionary process, to estimate the COCOMO model parameters are given in the Table 3.

Model 1

Experiment (without bias Function): The following are the results obtained by running the above algorithm implemented in "C". Totally we consider 20 projects. Number of iterations 100, Number of particles considered are 50.

a = 2.655034, b = 1.120601. The range of a is [1, 10] and b is [-5,5].

Model 2

Experiment (with bias Function): The following are the results obtained by running the above algorithm implemented in "C". Totally we consider 20 projects. Number of iterations 100, Number of particles considered are 50.

a = 1.538113, b = 1.270503 and c = 2.800148. The range of a is [1, 10] b is [-5,5] and c is [-5, 5]

Table 3: NASA software projects data (Prasad Reddy, 2010)

Project No.	Size	Effort adjustment factor	Measured effort
1	46.00	1.17	240.0
2	16.00	0.66	33.0
4	6.90	0.40	8.0
10	24.00	0.85	79.0
14	1.90	1.78	9.0
21	2.14	1.00	7.3
22	1.98	0.91	5.9
28	34.00	0.34	47.0
30	6.20	0.39	8.0
31	2.50	0.96	8.0
32	5.30	0.25	6.0
33	19.50	0.63	45.0
38	23.00	0.38	36.0
42	8.20	1.90	41.0
43	5.30	1.15	14.0
44	4.40	0.93	20.0
49	21.00	0.87	70.0
51	28.00	0.45	50.0
52	9.10	1.15	38.0
53	10.00	0.39	15.0

RESULTS AND DISCUSSION

The COCOMO model was provided by Boehm (1981) was given in Eq. 1. Our goal is to use PSO with constriction factor to tune the simple COCOMO model parameters such that better software effort estimation is achieved. It is tested on COCOMO dataset provided by Bohem. Table 4 shows the actual measured efforts of our models and we explored various model estimate the software effort for all given projects using Halsted, Walston-Felix, Bailey-Basili, Doty (Sheta *et al.*, 2008), standard COCOMO, Triangular member function and Gbell Function (Prasad Reddy, 2010).

Comparison with other models: Prasad Reddy(2010) provided a Triangular Membership function and Gbell function calculation, (Sheta *et al.*, 2008) provided Halsted, Walston-Felix, Bailey-Basili, Doty formulas for effort estimation. Typical models for software effort estimation are as shown below (Sheta *et al.*, 2008):

$$\text{Halstead } E = 5.2(\text{KLOC})1.50 \tag{11}$$

$$\text{Walston-Felix } E = 0.7(\text{KLOC})^{0.91} \tag{12}$$

$$\text{Bailey-Basili } E = 5.5 + 0.73(\text{KLOC})^{1.16} \tag{13}$$

$$\text{Doty (for KLOC} > 9) \; E = 5.288(\text{KLOC})^{1.047} \tag{14}$$

The measured effort and estimated efforts of various models are given in Table 4 and shows our model results are very close to the measured effort.

Table 4: Measured effort vs estimated efforts of various models

Project No.	Size	Effort adjustment factor	Measured effort	Halstead model	Walston flexi	Bailey basili	Doty model	Cocomo effort	Effort using TMF	Effort using G Bell	Estimated effort Model-1	Estimated effort Model-2
1	46.00	1.17	240.0	1622.33	22.81	67.46	291.21	212.0	246	252	226.75	235.99
2	16.00	0.66	33.0	332.80	8.73	23.70	96.38	39.0	41	41	39.17	37.19
4	6.90	0.40	8.0	94.25	4.06	12.36	39.95	9.8	11	11	9.25	9.96
10	24.00	0.85	79.0	611.39	12.62	34.63	147.36	108.0	138	138	79.46	76.93
14	1.90	1.78	9.0	13.62	1.26	7.04	10.35	10.7	10	9	9.70	8.99
21	2.14	1.00	7.3	16.28	1.40	7.26	11.73	7.0	7	7	6.23	6.84
22	1.98	0.91	5.9	14.49	1.30	7.11	10.81	5.8	6	6	5.19	6.13
28	34.00	0.34	47.0	1030.91	17.33	49.14	212.20	44.0	46	47	46.96	48.95
30	6.20	0.39	8.0	80.28	3.68	11.56	35.72	8.4	9	9	8.00	8.89
31	2.50	0.96	8.0	20.55	1.61	7.61	13.80	8.9	9	9	7.12	7.53
32	5.30	0.25	6.0	63.45	3.19	10.55	30.31	4.7	5	5	4.30	6.00
33	19.50	0.63	45.0	447.77	10.45	28.40	118.57	46.0	48	49	46.67	45.00
38	23.00	0.38	36.0	573.58	12.14	33.23	140.94	33.0	35	35	33.87	34.19
42	8.20	1.90	41.0	122.10	4.75	13.88	47.87	55.0	61	61	53.31	45.14
43	5.30	1.15	14.0	63.45	3.19	10.55	30.31	22.0	23	23	19.79	17.52
44	4.40	0.93	20.0	47.99	2.70	9.57	24.95	14.0	16	16	12.99	12.20
49	21.00	0.87	70.0	500.42	11.18	30.45	128.13	68.0	72	72	70.03	66.83
51	28.00	0.45	50.0	770.44	14.52	40.34	173.17	47.0	50	50	50.00	50.53
52	9.10	1.15	38.0	142.75	5.22	14.96	53.38	42.0	40	40	36.26	32.05
53	10.00	0.39	15.0	164.44	5.69	16.05	58.92	17.0	15	15	13.67	13.98

PERFORMANCE ANALYSIS

The three criterions are:

- Variance Accounted – For (VAF):

$$\%\,\mathrm{VAF} = \left[1 - \frac{\mathrm{var\ (Measured\ Effort\ -\ Estimated\ Effort)}}{\mathrm{var\ (measured\ effort)}}\right] \times 100$$

- Mean Absolute Relative Error (MARE):

$$\%\,\mathrm{MARE} = \mathrm{mean}\left[\frac{\mathrm{abs\ (Measured\ \ Effort\ -\ Estimated\ Effort)}}{\mathrm{(measured\ effort)}}\right] \times 100$$

- Variance Absolute Relative Error (VARE):

$$\%\,\mathrm{VARE} = \mathrm{var}\left[\frac{\mathrm{(abs\ (Measured\ Effort\ -\ Estimated\ Effort)}}{\mathrm{(measured\ effort)}}\right] \times 100$$

Present intention concern the development of model structures which can generalize the effort, computed for all projects under study. PSO algorithms were used to estimate the standard COCOMO variable model parameters. Two models are provided. It can be seen that the PSO with constriction factor for COCOMO 81 model outperform the Halsted, Walston-Felix, Bailey-Basili, Doty, standard COCOMO, Triangular member function and GBell Function. The computed values of MARE, VAF and VARE for all models are given in Table 5.

Table 5: Performance analysis

Model	Variance accounted for	Mean absolute relative error	Variance absolute relative error
	(%)		
Halstead	5106.36	703.7313	3044.758
Walston-Felix	17.66	74.2187	1.753
Bailey-Basili	45.77	35.7163	6.140
Doty	25.79	165.4853	190.364
COCOMO Effort	97.09	16.1306	2.049
Effort Using TMF	93.50	17.51334	4.832
Effort Using Gbell	93.40	17.08752	4.948
Estimated Effort Model-1	99.42	12.1920	1.570
Estimated Effort Model-2	99.82	9.0143	1.037

CONCLUSION

In this study we have introduced an important generalization of the COCOMO software cost estimation model by augmenting with particle swarm optimization with constriction factor. This study is based on 53 projects, among we taken 20 projects for training and testing. The analysis based on VAF, MARE and VARE shows that PSO with constriction factor always leads to a satisfactory result. The obtained results are superior as compared to previously reported work in the literature. This work can be applied to other models of Software Cost Estimation.

REFERENCES

Bailey, J.W. and V.R. Basili, 1981. A meta model for software development resource expenditures. Proceedings of the 5th International Conference on Software Engineering, (ICSE'81), IEE Explore, pp: 107-129.

Boehm, B.W., 1981. Software Engineering Economics. Prentice-Hall, Englewood Cliffs, New Jersey, ISBN: 0138221227, pp: 767.

Clerc, M., 1999. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. Proceedings of the Congress on Evolutionary Computation, (CEC'99), Washington, DC., pp: 1951-1957.

Heng, X.C., Z. Qin, X.H. Wang and L.P. Shao, 2006. Research on learning bayesian networks by particle swarm optimization. Inform. Technol. J., 5: 540-545.

Huang, S.J., C.V. Lin and N.H. Chiu, 2006. Fuzzy decision tree approach for embedding risk assessment information into software cost estimation model. J. Inform. Sci. Eng., 22: 297-313.

Idri, A., A. Abran and S. Mbarki, 2006. An experiment on the design of radial basis function neural networks for software cost estimation. Proceedings of the Information and Communication Technologies, (ICT'06), Damascus, pp: 1612-1617.

Iraj, H. and M. Saleh, 2008. PSO-based controller design for rotary inverted pendulum system. J. Applied Sci., 8: 2907-2912.

Jorgensen, M. and M. Shepperd, 2007. A systematic review of software development cost estimation studies. IEEE Trans. Software Eng., 33: 33-53.

Kaur, K. and H. Singh, 2010. Candidate process models for component based software development. J. Software Eng., 4: 16-29.

Poli, R., J. Kennedy and T. Blackwell, 2007. Particle swarm optimization an overview. Swarm Intell., 1: 33-57.

Prasad Reddy, P.V.G.D., 2010. Particle swarm optimization in the fine-tuning of fuzzy software cost estimation models. Int. J. Software Eng., 1: 12-22.

Sengupta, G.J., 2010. Regression testing method based on XML schema for GUI components. J. Software Eng., 4: 137-146.

Sheta, A.F., 2006. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. J. Comput. Sci., 2: 118-123.

Sheta, A., D. Rine and A. Ayesh, 2008. Development of software effort and schedule estimation models using soft computing techniques. Proceedings of the IEEE Congress on Evolutionary Computation, June 1-6, Hong Kong, pp: 1283-1289.

Sutha, S. and N. Kamaraj, 2008. Particle swarm optimization applications to static security enhancement using multi type facts devices. J. Artif. Intell., 1: 34-43.

Zeng, J. and Z. Cui, 2007. Differential Meta-Model and Particle Swarm Optimization. I-Tech Education and Publishing, Vienna, Austria, pp: 101-112.