Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 9, No. 1, February 2018), Pages: 41-49
www.jacr.iausari.ac.ir

An Improved Algorithmic Method for Software Development Effort Estimation

Elham Khatibi, Vahid Khatibi Bardsiri✉
Department of Computer Engineering, Faculty of Science, Kerman Branch, Islamic Azad University, Kerman, Iran
kbelham2@live.utm.my; kvahid2@live.utm.my

Received: 2017/01/07; Accepted: 2017/04/06

Abstract

Accurate estimating is one of the most important activities in the field of software project management. Different aspects of software projects must be estimated among which time and effort are of significant importance to efficient project planning. Due to complexity of software projects and lack of information at the early stages of project, reliable effort estimation is a challenging issue. In this paper, a hybrid model is proposed to estimate the effort of software projects. The proposed model is a combination of particle swarm optimization algorithm and a linear regression method in which coefficient finding is optimally performed. Moreover, the estimation equation is adjusted using project size metric so that the most accurate estimate is achieved. A relatively real large data set is employed to evaluate the performance of the proposed model and the results are compared with other models. The obtained results showed that the proposed hybrid model can improve the accuracy of estimates.

Keywords: *Effort Estimation, Particle Swarm Optimization Algorithm, Software Project, Linear Regression*

1. Introduction

Software developers and researchers normally use various techniques to estimate the software development effort. Accurate estimation of costs is required to determine the time and human resource in software projects. Wrong estimation may sometimes lead to unpredictable results and project failure. Various estimation models have been proposed for effort estimation during the last decade. However, there hasn't been an agreement on selecting the best estimation model so that none of existing models has been suitable for all projects and databases. The first idea for software effort estimation returns to 1950 by presenting the manual rule of thumb [1]. By increasing the number of software projects and need of user society to earn high quality software, some models based on the linear equations were presented as the software effort techniques in 1965[2]. As the pioneers of software estimation methods, we can consider the name of Larry Putnam, Barry Bohem and Joe Aron [1]. Afterward, in 1973, the IBM researchers presented the first automated tool, Interactive Productivity and Quality (IPQ) [1]. Barry Boehm proposed a new method based on computing some of software project factors by means of several mathematical equations called COCOMO [3]. In addition, Boehm explained several algorithms in his book "Software Engineering Economics" [3] that are still used

by researchers. The other models such as Putnam Lifecycle Management (SLIM) [4] and Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) continued the principals of COCOMO[2]. Introducing the Function Point (FP) as a metric for software size estimation by Albrecht [5] was the other important event in that decade .

Optimization algorithms have been widely used in the field of software development effort estimation. Genetic algorithm [6-8], particle swarm optimization algorithm [9-10], firefly optimization algorithm [11] and ant colony optimization algorithm [12] are instances of algorithms utilized by researchers in this area. In spite of using a wide range of optimization algorithms, the accuracy of estimates is not satisfying. On the other hands, the existing optimization based methods are mostly focused on non-algorithmic estimators. In this paper, a hybrid model is proposed in which the regression and PSO algorithms are combined.

This paper is organized in 6 sections. Sections 2 and 3 are related to explanation of PSO and linear regression, respectively. Section 4 includes the details of the proposed model. In Section 5, the experimental results are shown and elaborated. Conclusion and future work are explained in Section 6.

2. Particle Swarm Optimization (PSO) Algorithm

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [13], inspired by social behavior of bird flocking or fish schooling. The main strength of PSO is its fast convergence, which is comparable to many global optimization algorithms like Genetic algorithms, Simulated Annealing and other global optimization algorithms. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. The detailed information will be given in following sections.

2.1 Optimization Process

PSO learns from the scenario and uses it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. It is called "particle". All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. In any iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) particle has achieved so far (The fitness value is also stored). This value is called pBest. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and called gBest. After finding the two best values, the particle updates its velocity and positions with following Equations 1 and 2.

$$v[] = w * v[] + c_1 * rand() * \left(pBest[] - present[]\right) + c_2 * rand() * \left(gbest[] - present[]\right) \quad (1)$$

$$present[] = persent[] + v[] \quad (2)$$

where, v[] is the particle velocity, persent[] is the current particle (solution). Pbest[] and gbest[] are defined as stated before. rand () is a random number between (0,1). The term c_1 is the "cognitive parameter", the term c_2 is the "social parameter" and term w is called inertia weight. The combination of these parameters determines the convergence properties of the algorithm.

2.2 Parameter Values Selection

For the initial version of the PSO, the values for c_1, c_2 and w have to be selected. This selection has an impact on the convergence speed and the ability of the algorithm to find the optimum, but different values may be better for different problems. Much work has been done to select a combination of values that works well in a wide range of problems.

Regarding the inertia weight, it determines how the previous velocity of the particle influences the velocity in the next iteration:

- If w=0, the velocity of the particle is only determined by the $part_i$ and $part_g$ positions; this means that the particle may change its velocity instantly if it is moving far from the best positions in its knowledge. Thus, low inertia weights favor exploitation (local search).
- If w is high, the rate at which the particle may change its velocity is lower (it has an "inertia" that makes it follow its original path) even when better fitness values are known. Thus, high inertia weights favor exploration (global search).

In [14-15], a decaying inertia weight is proposed and tested, with the aim of favoring global search at the start of the algorithm and local search later. If the inertia weight is not reduced with time, they suggest selecting a value of $w \in [0.8, 1.2]$. The values of the cognitive and social factors are not critical for the algorithm, but selection of proper values may result in better performance, both in terms of speed of convergence and alleviation of local minima. Their values have to be taken into account when choosing the inertia weight. Several studies propose different values for these parameters that are considered adequate for some of the usual benchmark functions. The pseudo code of the procedure is as follows:

For *each particle*
  Initialize particle
End For
Do
For *each particle*
  Calculate fitness value
  If *the fitness value is better than the best fitness value (pBest) in history Then*
    Set current value as the new pBest
  End If
End For
Choose the particle with the best fitness value of all the particles as the gBest
For *each particle*
  Calculate particle velocity according to Equation (1)
  Update particle position according to Equation (2)
End For
While *maximum iterations or minimum error criteria is not attained*

Particles' velocities on each dimension are clamped to a maximum velocity Vmax. If the sum of accelerations would cause the velocity on that dimension to exceed Vmax, which is a parameter specified by the user, then the velocity on that dimension is limited to Vmax.

3. Linear Regression

In statistics, linear regression is an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of y given the value of X is assumed to be an affine function of X. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear regression has many practical uses. Most applications fall into one of the following two broad categories:

If the goal is prediction, or forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of y and X values. After developing such a model, if an additional value of X is then given without its accompanying value of y, the fitted model can be used to make a prediction of the value of y.

Given a variable y and a number of variables X_1, ..., Xp that may be related to y, linear regression analysis can be applied to quantify the strength of the relationship between y and the Xj, to assess which Xj may have no relationship with y at all, and to identify which subsets of the Xj contain redundant information about y.

4. Proposed Model

In this section, a novel model is proposed to increase the accuracy of software development effort estimation by combining linear regression model and PSO algorithm. The complicated relation between the software project attributes and the required effort is the main problem to reach accurate and reliable estimates. This relation is quite different from one data set to another one due to the nature of software projects. Numerous factors can affect the process of software project development so that project managers must deal with the contradictory situations to estimate the required effort in early stages of project.

Therefore, a flexible model is required to handle this complexity. The problem of software development effort estimation has attracted the attention of researchers in recent years. The main focus of prior studies is finding a flexible and accurate equation to create a relation between dependent variable (effort) and independent variables. Non-algorithmic methods such as analogy based estimation, neural network and expert judgment have widely been studied in this area.

4.1 Performance Metrics

In order to construct the proposed hybrid model, performance metrics must be elaborated. The estimator performance is evaluated using standard metrics including RE (Relative Error), MRE (Magnitude of Relative Error), MMRE (Mean Magnitude of Relative Error) and MdMRE (Median of MRE) which are computed as follows [16] .

$$RE = \left(Estimated \ - \ Actual\right) / \left(Actual\right) \tag{3}$$

$$MRE = \left|Estimated - \ Actual \ \right| / \left(Actual\right) \tag{4}$$

$$MMRE = \sum MRE \ / \ N \tag{5}$$

$$MdMRE = Median\left(MREs\right) \tag{6}$$

The other parameter used to evaluate of performance is PRED (Percentage of the Prediction) which is determined as follows:

$$PRED\left(X\right) = A / N \tag{7}$$

Where, A is the number of projects having MRE less than or equal to X while N is the number of projects. Normally, the acceptable level of X in software development effort estimation methods is 0.25. Decreasing of MMRE and increasing of PRED are the main aims of all estimation techniques in this field.

4.2 Optimization Process

The proposed model is a simple linear regression defined between actual effort and other attributes of project. Equation 8 shows the proposed model.

$$Est_j = \left(\sum_{i=1}^{N} a_i \times c_i\right) + adj \tag{8}$$

Where, Est_j is the final estimate of effort for project j and a_i is the ith attribute of project j. The main goal is finding the most suitable coefficients c_i for attributes. Indeed, the final model is constructed using the coefficients found by particle swarm optimization algorithm. Moreover, an adjustment factor (adj) is defined to make the model flexible enough. The adj parameter is computed by PSO using the size metrics. FP and LOC are the most common size attributes in effort estimation data sets. Therefore, in the proposed model, these attributes are utilized to compute adj as follows:

$$adj = size^r \tag{9}$$

Where, size can be determined by FP or LOC and r is computed by PSO as an unknown parameter. This form of equation is selected because it has been widely used in prior studies. Therefore, PSO must compute N parameters including N-1coefficients

for non-size attributes and one more parameter (r) as exponent of size attribute. In other words, non-size attributes are separated from size attribute and PSO makes a combined equation.

4.3 Data set

In order to investigate the performance of the proposed model, a real data set is employed. Dasharnais [17] is one of the most common data sets in the field of software effort estimation. Although this dataset is relatively old, it has been widely employed in many of recent research studies [18-20]. Table 1 provides the statistical information about this data set.

Table 1: Description of Desharnais dataset

Attribute	Description	Min	Max	Mean	Median
TeamExp	Team experience	0	4	2.30	2
ManagerExp	Manager's Experience	0	7	2.65	3
Length	Length of project	1	36	11.30	10
Transactions	Number of transactions	9	886	177.47	134
Entities	Number of entities	7	387	120.55	96
AdjustFactor	Sum of complexity factors	5	52	27.45	28
PointsAdjust	Number of adjusted function points	73	1127	298.01	247
Language	Programming language	1	3	1.56	1
Effort (1000h)	Development effort	0.55	23.94	4.83	3.54

In this data set, there are 81 projects related to a Canadian software company, out of which four projects include missing values and the remaining 77 projects are considered in the evaluation process.

5. Numerical Results

In order to evaluate the performance of the proposed model, PSO algorithm is configured as shown in Table 2. These parameters were found through an exhaustive process of trial and error. Regarding the parameters of C_1, C_2 and W, the values used in this paper is the same as what utilized in prior studies.

Table 2: Initial setting of PSO

Feature	Value
Population	200
Variables	8
Iteration	350
C_1	2
C_2	2
W	1

Table 3 shows the results of the proposed model on Desharnais data set based on MMRE and PRED(0.25). The results are presented using three-fold cross validation method. 30 times of independent executions were conducted and the average of results was considered as the final accuracy.

Table 3: Performance metrics on Desharnais data set

	Fold1	Fold2	Fold3	Average
MMRE	0.63	0.76	0.32	0.57
PRED	0.23	0.21	0.54	0.33

In order to compare the performance of the proposed model with the other models, multiple linear regression (MLR) and genetic algorithm (GA) were employed. Mutation and crossover rate in GA are 0.02 and 0.8, respectively. Moreover, population and iterations parameters are set to 300 and 400, respectively. Figure 1 depicts the comparison of MLR, GA and the proposed model in MMRE and PRED(0.25).

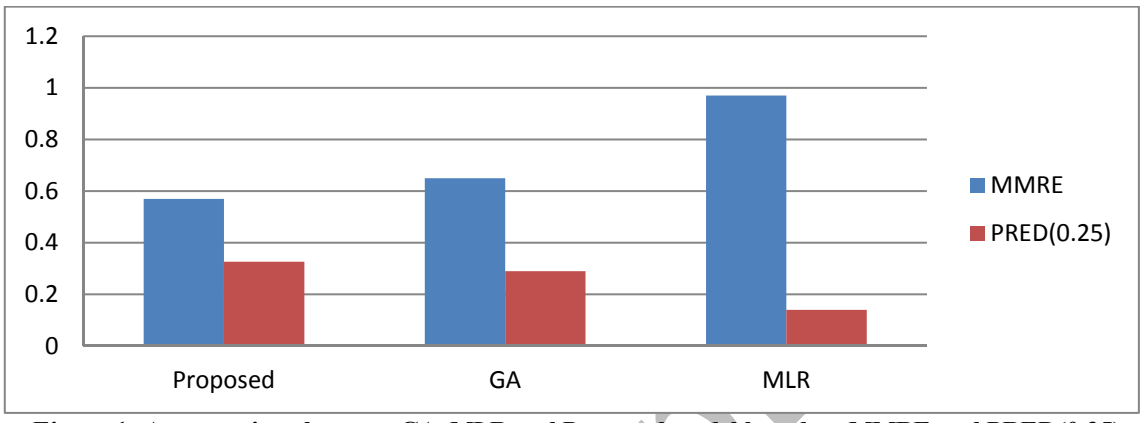

Figure 1: A comparison between GA, MLR and Proposed model based on MMRE and PRED(0.25)

It is observed that the proposed model can outperform GA and MLR in both MMRE and MLR. This comparison approves the capability of PSO algorithm in optimization of linear coefficients in the field of software development effort estimation. The role of adjusting and coefficient finding is obviously seen in the figure.

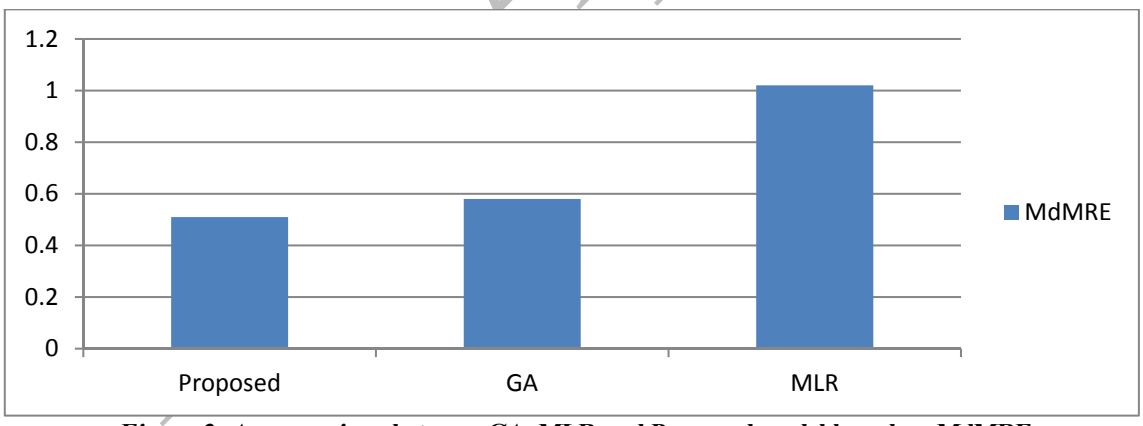

Figure 2: A comparison between GA, MLR and Proposed model based on MdMRE

In addition to increasing the accuracy of estimates in terms of MMRE and PRED(0.25), the parameter of MdMRE is also improved, as seen in Figure 2. It is observed that the value of MdMRE for the proposed model is less than that of GA and MLR, which indicates the superiority of the proposed model in this parameter.

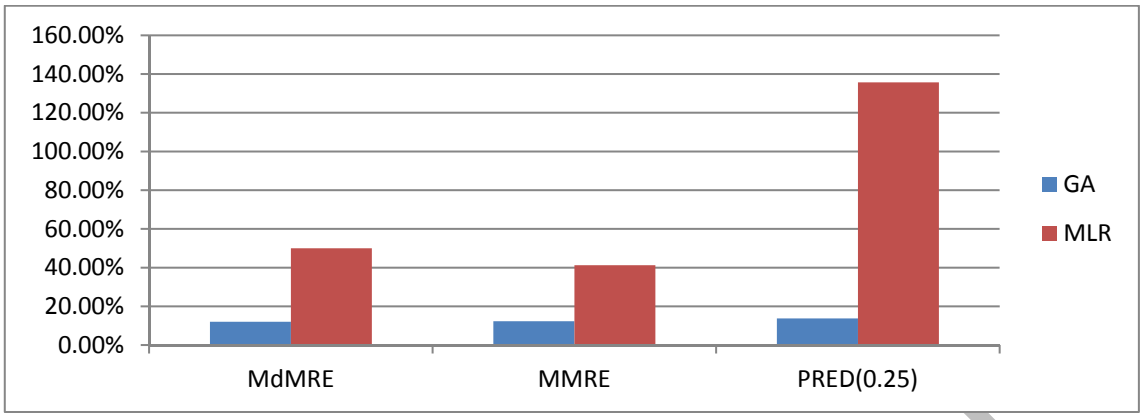

Figure 3: Improvement analysis based on performance metrics

Finally, improvement percentage obtained by the proposed model in terms of three performance parameters is seen in Figure 3. It is clearly seen that the proposed model has significantly improved the performance metrics. Particularly, the performance of MLR has considerably improved by the proposed model, which shows the capability of PSO to adjust the coefficients.

6. Conclusion

The estimation of effort required to conduct and complete a software project is very important for project managers. The complicated and uncertain nature of software projects makes the estimation process difficult and vague. However many estimation models have been suggested in this field, the accuracy level is not satisfying yet. In this paper, a combination of linear regression and PSO algorithm was proposed to increase the accuracy of estimates. The proposed model used an adjustment factor along with the common coefficients to make the estimation process more flexible and accurate.

A real data set including 77 software projects was employed to evaluate the performance of the proposed model. Multiple linear regression and genetic algorithm were compared with the proposed model. In performance metrics of MMRE, MdMRE and PRED(0.25), the proposed model outperforms GA and MLR. The promising results indicate that PSO can be a suitable selection for optimization of linear equations defined for software development effort estimation. As the future work, the proposed model will be strengthened with feature selection techniques.

References

[1] C. Jones, *Estimating software costs: Bringing realism to estimating*, 2nd ed. New York: NY: McGraw-Hill, 2007.

[2] B. W. Boehm and R. Valerdi, "Achievements and Challenges in Cocomo-Based Software Resource Estimation," *IEEE Softw.,* vol. 25, pp. 74-83, 2008.

[3] B. W. Boehm, *Software engineering economics*. Englewood Cliffs: NJ: Prentice Hall, 1981.

[4] L. H. Putnam, "A general empirical solution to the macrosoftware sizing and estimating problem," *IEEE Transactions on Software Engineering,* vol. 4, pp. 345-361, 1987.

[5] A. J. Albrecht and J. A. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation," *IEEE Trans Software Eng. SE,* vol. 9, pp. 639-648, 1983.

[6] S.-J. Huang, *et al.*, "Integration of the grey relational analysis with genetic algorithm for software effort estimation," *European Journal of Operational Research,* vol. 188, pp. 898-909, 2008.

[7] A. L. I. Oliveira, *et al.*, "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.

[8] K. K. Shukla, "Neuro-genetic prediction of software development effort," *Information and Software Technology,* vol. 42, pp. 701-713, 2000.

[9] Y.-L. Huang, *et al.*, "An Improved forecasting model based on the weighted fuzzy relationship matrix combined with a PSO adaptation for enrolments," *International Journal of Innovative Computing, Information and Control* vol. 7, pp. 4027-4045, 2011.

[10] V. Khatibi Bardsiri, *et al.*, "A PSO-based model to increase the accuracy of software development effort estimation," *Software Quality Journal,* vol. 21, pp. 501-526, 2013.

[11] N. Ghatasheh, *et al.*, "Optimizing Software Effort Estimation Models Using Firefly Algorithm," *Journal of Software Engineering and Applications,,* vol. 8, pp. 133-142, 2015.

[12] N. Akhtar, "Perceptual Evolution for Software Project Cost Estimation using Ant Colony System," *International Journal of Computer Applications,* vol. 81, pp. 23-30, 2013.

[13] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942–1948, 1995.

[14] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *the Seventh Annual Conference on Evolutionary Programming*, pp. 591-600, 1998..

[15] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *IEEE Congress on Evolutionary Computation*, pp. 1945-1950, 1999.

[16] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies," *IEEE Transaction on software engineering,,* vol. 23, pp. 736-743, 1997.

[17] J. Desharnais, "Analyse statistique de la productivitie des projets informatique a partie de la technique des point des foncti on," Master of Science, University of Montreal, 1989.

[18] M. Auer, *et al.*, "Optimal project feature weights in analogy-based cost estimation: improvement and limitations," *IEEE Transactions on Software Engineering,* , vol. 32, pp. 83-92, 2006.

[19] Y. F. Li, *et al.*, "A study of the non-linear adjustment for analogy based software cost estimation," *Empir Software Eng,* vol. 14, pp. 603-643, 2009.

[20] P. Jodpimai, *et al.*, "Estimating Software Effort with Minimum Features Using Neural Functional Approximation," in *International Conference on Computational Science and Its Applications (ICCSA)*, Brazil, pp. 266-273, 2010.

[21] Bansal, A.J.i.R.G.A., Ranking of Software Effort Estimation Selection Criteria Based On Fuzzy Set Theory. *International Journal of Advance Research and Innovation*, vol.4. pp. 43-47, 2016.