TELKOMNIKA, Vol.16, No.5, October 2018, pp.2208~2216
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v16i5.9703

Optimizing Effort Parameter of COCOMO II Using Particle Swarm Optimization Method

Kholed Langsari[1], Riyanarto Sarno*[2], Sholiq[3]
[1]Fatoni University, Thailand
[1,2]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
[3]Department of Information Systems, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
*Corresponding author, e-mail: langsaree@gmail.com[1], riyanarto@if.its.ac.id[2], sholiq@is.its.ac.id[3]

Abstract

Estimating the effort and cost of software is an important activity for software project managers. A poor estimate (overestimates or underestimates) will result in poor software project management. To handle this problem, many researchers have proposed various models for estimating software cost. Constructive Cost Model II (COCOMO II) is one of the best known and widely used models for estimating software costs. To estimate the cost of a software project, the COCOMO II model uses software size, cost drivers, scale factors as inputs. However, this model is still lacking in terms of accuracy. To improve the accuracy of COCOMO II model, this study examines the effect of the cost factor and scale factor in improving the accuracy of effort estimation. In this study, we initialized using Particle Swarm Optimization (PSO) to optimize the parameters in a model of COCOMO II. The method proposed is implemented using the Turkish Software Industry dataset which has 12 data items. The method can handle improper and uncertain inputs efficiently, as well as improves the reliability of software effort. The experiment results by MMRE were 34.1939%, indicating better high accuracy and significantly minimizing error 698.9461% and 104.876%.

Keywords*: particle swarm optimization*, *estimation of software effort, COCOMO II*

1. Introduction

The cost estimated of software is one of the major challenges in the management of software project development. The accuracy of software cost estimation is very important for use as a reference by the project manager for managing activities related to Software Development Life Cycle (SDLC). The main tasks of software project managers are ensuring that the project is achieved with the aim of "high-quality software must be produced at a low cost of concern in time and budget". In addition, good managers of a software project can appropriately forecast the cost and others resources of the project which they handle. Before getting project costs, it is usually determined the effort required to complete the software development project. An effort is expressed in a person-month. Activity estimates of costs and efforts are intended to obtain an accurate estimation result, not the result of either overestimate or underestimate. It can also manage the application's quick configuration [1]. The accuracy is determined by several variables or cost drivers of a method that is used, so to get an accurate cost estimate of the software requires control of the variables or cost drivers that affect it.

Estimating software costs in the early phases can increase control of project activities that is planning, budgeting, and monitoring. An appropriate estimate can prepare well to espouse for the process of decision-making, as well as all processes in a project of software development can be safeguarded efficiently and effectively, because for a project generally has limited resources. The main difficult problems for estimating software costs are both inherent uncertainty in software development and the complex and dynamic interplay of factors influencing the software development effort used. There are both several techniques and procedures for dealing with the problem. Both methods are algorithmic and non-algorithmic which can be used to forecast software cost [2,3]. The technique of algorithmic refers to the existence of a mathematical equation to estimate software costs, whereas non-algorithmic techniques refer to the absence of mathematical equations to estimate software costs [3]. This

Received April 27, 2018; Revised June 7, 2018; Accepted July 14, 2018

includes the analogy method [4], artificial neural network [5,6], fuzzy logic [7,8], genetic algorithm [9], and Cuckoo optimalization [10].

The cost estimated is expressed in terms of currency ($). Before getting the cost estimated of software development projects, it is usually preceded by estimating effort which is expressed by the amount of person-time to finish a project. Generally, more effort is used, more cost that is spent.

Several methods of cost estimation in recent decades have been introduced and proposed by many researchers. Among all the software cost estimation methods, the COCOMO is the most well-known and widely applied in calculating software costs [11]. Problems come on regarding the exactitude of the implementation of the method to complete the software cost estimated. Techniques of heuristic are used to resolve the limitations of this method and refine its application [12]. A variety of methods in heuristic optimization are used in optimization issues. This method can be applied for software cost estimation as well. These methods are Particle Swarm Optimization [13-17], Genetic Algorithm [9,18], Firefly Algorithm [19], and others.

This paper provides the study of PSO as an algorithm of optimization. It is used to optimize the parameters in the model of COCOMO II, so resulting in more accurate and precise efforts, cost, and time of development. The remaining sections of this paper are composed in the following manner: Section 2 briefly describes the related work already investigated to estimate the effort through different methods and with the PSO approach. Section 3 describes a working methodology consisting of the steps used in this experiment. Section 4 presents the results already obtained and analyzes the discussion of the results. Section 5, the final part concludes this research that the accuracy of the estimates can be refined with the adoption of the PSO approach.

2. Related Work

The calibration of COCOMO II coefficients to optimize and improve the accuracy of estimation results has been proposed by some previous research. Sarno and Sidabutar [5] investigated the role of software sizes stated in the Line of Code (LOC) and Effort Multiplier (EM) to increase the accuracy of estimate result. Fuzzy Logic using Gaussian Membership Function (GMF) is implemented to COCOMO II for EM. The GMF can make a smoother transition that it means more accuracy of effort multipliers. Also, they implemented the Neural Network (NN) as an approach applied a feed-forward neural network of multi-layer using a back-propagation learning algorithm [6]. The model offered provides significantly more improvement than the basis fuzzy model or the original COCOMO model. For local calibration using models referred to the Fuzzy Logic and Tabu Search approach has been proposed by Baiquni, et al. [20]. They refine grade of precision by obscuring cost drivers in the COCOMO II using Gaussian Membership Functions (GMF) of Fuzzy Logic for redesigning EM. To find new parameter values in COCOMO II model is used local calibration apply Calico and Tabu Search. The new value found can significantly refine precision or degrade errors.

Optimizing the coefficient of COCOMO II model with dataset of NASA using PSO techniques has been proposed by Parkas and Kamabir [15]. In their study, it was found that optimization problems could be solved efficiently and uncertainties could be reduced better using PSO than using original coefficient values. Likewise, Kumar et.al [13] and Sheta et.al [19] have analyzed for PSO optimization along with both Linear regression and Fuzzy Logic by composing a collection of linear models to degrade errors of cost estimation. The NASA18 data set is used on COCOMO II models in their research.

The PSO provides an efficient technique for optimizing estimation of effort, while the method of linear regression gives good results but will take time. Reddy et.al [14] initiated the significant generalization and introduced new models by adding PSO using Constriction Factor for tuning parameters of COCOMO II. This new model can handle uncertain and improper inputs and it can improve the reliability of cost estimated. Experiments conducted by the researchers showed that PSO with a tightening factor gave a satisfactory result. Reddy et al. [14] offered Multi-Objective Particle Swarm Optimization (MOPSO) as a new model for the cost estimated of software. According to their study that the model has given better results when it is compared with the original COCOMO II.

2.1. Cost Constructive Model (COCOMO) II

Several models of software cost estimation have been offered and promoted to assist in providing accurate forecasts to assist managers of a project in making correct decisions about their projects [5]. One of the most famous and widely used models of effort estimated is the Constructive Cost Model (COCOMO) which was first introduced by Barry Boehm in 1981 [21]. As an estimate of effort, schedule, cost of planning the process of software development activities used COCOMO as a model. This model was constructed from 63 items of data in a software project dataset in which each data item consist of sixteen variables (cost drivers). Cost Drivers in COCOMO is categorized into three aspects such as Line of Code (LOC), Scale Factors (SF), and Effort Multiplier (EM). All cost drivers generated effort in person-month (PM). COCOMO II was introduced by Barry Boehm in 2000 as a model which has been supplied more accurately with some aspects of improvement in some cost drivers.

The COCOMO II includes several software attributes such as 17 Effort Multipliers (EM), 5 Scale Factors (SF), Software Size (in KLOC), and the effort estimated which are used in the COCOMO II Architecture Post Model. Multiplier attempts are grouped into four categories and there are 5 Factor Scales (SF).

$$Effort(PM) = A \cdot Size^{E} \times \prod_{i=1}^{17} EM_i + PM_{Auto} \quad (1)$$

In the model of COCOMO II [4], the equations which are used to calculate software development efforts are shown in equation (1). Where, A is a multiplication constant, has a value of 2.94 that measures effort according to a particular project condition. Size is defined as the size estimated of software in Kilo Source Lines of Code (KSLOC). The E is scale expansion for effort. It is the factor of exponential that has the account record for the relative scale of economies or diseconomies deal with correcting for the size of software projects increasing, and EM_i is the Efforts Multiplier in which i=1, 2, 3,4 ....17. Computing the Scale Factor, the coefficient of E is determined by the equation (2):

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j \quad (2)$$

where B is a constant of exponential holding a value of 0.91 and SF_j is a Scale Factor where j=1, 2, 3, 4, or 5. This paper attempts by optimizing the two parameters in the model of COCOMO II, the constant of multiplier A and the constant of exponential B.

2.2. Particle Swarm Optimization (PSO)

PSO referred on swarm behavior in nature, such as schools of fish and birds called swarm intelligence. PSO was introduced and developed by Kennedy and Eberhart in 1995 [22] and has become one of the most widely used intelligence-based algorithms due to its simplicity and flexibility. Instead of using mutations or crossovers or pheromones, it applies randomness and communication in global among swarm particles [23].

The algorithm of PSO seeks the area of the objective function by updating the path of each agent, named the particle, such the connection path formed by the quasi-stochastic position vector [6,23]. The clumped particles movement consists of two main components: the stochastic component and the deterministic component. Each particle is attracted to the best global position right now g* from its best location x_i* in its history, while at the same time the tendency to move randomly.

When a particle gets or finds a better location than the previously discovered location, it then updates the location as the best current for particle i. There is the best current for all particles n at any time during the iteration process. The purpose of this process is to get the best global solution among all the best solutions right now until the goal is no longer upgraded or after a certain iteration. Movement of the particle is schematically shown in Figure 1, where $x_i^{*(t)}$ is the best of the current for particles i, and $g^* \approx min\{f(x_i)\}$ for (i = 1, 2, ..., n) i is the best global current at t.

$$v_{ij}^{t+1} = wv_{ij}^{t} + c_1 r_1 \left[P_{best,i}^{t} - x_{ij}^{t}\right] + c_2 r_2 \left[G_{best}^{t} - x_{ij}^{t}\right] \quad (3)$$

where x_{ij} is vector positions and v_{ij} is vector velocities for particle i. Then, the new vector of velocity is updated using equation (3). Meanwhile, the initial location of all the particles must be uniformly distributed so that the particles can get samples in most places. The initial velocity of a particle can be given a value of zero, which is, $v^{t=0} = 0$. Foward, the new location may be regenerated with equation (4).

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1} \quad (4)$$

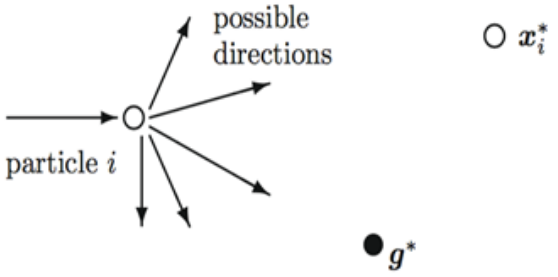

Figure 1. The particle motion schematic representation in the PSO moves towards the best global g* and the best current x_i for every particle i

Where x_{ij}^{t} is the recent search position, $x_{ij}^{t+1}$is an updated search position, v_{ij}^{t} is the current velocity, v_{ij}^{t+1} is the updated velocity, $P_{best,i}^{t}$ is the best experience of a particle, G_{best}^{t} is the best in the world, *w* is a weighted function, both r_1 and r_2 are two vectors at random, and each entry gets values between 0 and 1. The parameters c_1 and c_2 are acceleration constants or learning parameters, which can usually be taken as c1 ≈ c2 ≈ 2. In the technique of swarm optimization, looking for solutions in solution space in the range [-x, x]. Although v_i can have any value, it is usually limited in some range [0, v_{max}].

3. Research Method

In this paper, the COCOMO II model parameters are optimized using Particle Swarm Optimization. PSO is a good technique to solve the uncertainty of the data set and optimize the value that is relevant to the effort and relevant to show results with less time. PSO will take the minimum time and no need to predict the value. Minimum fitness is the initial value to start optimizing value, these values have two types, Pbest and Gbest. A collection of bunch iterations until the best show requires a fitness score. Each particle tries to modify and move its current position and speed according to the distance between the current position and Pbest, and the distance between the current position and Gbest. The inputs are software size, actual effort, EM, and SF, while the outputs are the values of parameters both A and B for local calibration value. The steps of the proposed Particle Swarm Optimization are:

a. Initialize the particle "n" with Pi's random position P_i and vector of velocity V_i of the optimization parameter. it also needs a speed range between [Vmin, Vmax]. The starting position of each individual particle is best (Pbest) for each Particle.
b. Initialize the value of weight function *w* with 1 parameter of weight and coefficient of personal acceleration c1, the social acceleration coefficient c2 with both standards is 2.0.
c. for i= 1, 2, 3, ..., n, for all particles and for every particle position by optimization parameter values, a function of fitness evaluation. The fitness function is the Mean Magnitude of Relative Error (MMRE) in equation (9) – equation (10) and Manhattan distance (MD) in equation (11). The goal is minimizing both MMRE and MD by selecting the suitable best value from the range stated in step 1.
d. Pbest is established for every particle to examine and to contrast the value of effort and effort estimated of the current and previous parameter values. If fitness (p) is better than fitness (Pbest) then set p as Pbest.

e. Set the best of Pbest as the best in global (Gbest). The value of particle for which variation between effort and effort estimated is the least selected as a Gbest particle.
f. Update velocity and position of the optimization based on equation (3) and (4). The new regulatory formula for parameter A in equation (5) and (6), similarly for parameter B in equation (7) and (8).

The equation for updating the velocity and position of parameter A is given as follows

$$v_{aj}^{t+1} = wv_{aj}^{t} + c_1 r_1 \left(P_{best,ij}^{t} - x_{aj}^{t}\right) + c_2 r_2 \left(G_{best,ij}^{t} - x_{aj}^{t}\right) \quad (5)$$

$$x_{aj}^{t+1} = x_{aj}^{t} + v_{aj}^{t+1} \quad (6)$$

Whereas the equations used to update the velocity and position of parameter B are also given as follows.

$$v_{bj}^{t+1} = wv_{bj}^{t} + c_1 r_1 \left(P_{best,ij}^{t} - x_{bj}^{t}\right) + c_2 r_2 \left(G_{best,ij}^{t} - x_{bj}^{t}\right) \quad (7)$$

$$x_{bj}^{t+1} = x_{bj}^{t} + v_{bj}^{t+1} \quad (8)$$

g. Give the best value as the optimal solution.
h. Repeat steps 3 through 7 up to the amount of user-defined iterations or particle conditions.

We promote using the Mean Magnitude of Relative Error (MMRE) and the difference between effort and estimate (Manhattan Distance or MD) like a function used as the offered method.

$$MRE_i = \frac{|Effort_i - Estimated\ Effort_i|}{Effort_i} \times 100\% \quad (9)$$

The key to the successful use of estimation methods that predicted results are more accurate than ever. The deviation ratio between actual efforts and efforts estimated should get the smallest value. The high difference between actual effort and effort estimated will have a meaningful impact the costs planning on projects of software development. In this study, we used MRE as common evaluation criteria in cost estimated of software for evaluating the accuracy of the expected effort. In equation (9), it is showed formula to calculate MRE for each observation (each project). The number of measurements of the accuracy level is formulated based on the evaluation criteria ie MRE which state the predictions individually. it can be averaged to produce Mean MRE (MMRE) [3] as stated in equation (10).

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} \frac{|Effort_i - Estimated\ Effort_i|}{Effort_i} \quad (10)$$

Manhattan Distance that calculates a completely different distance between effort and effort estimated The Manhattan distance is considered in equation (11).

$$MD = \left(\sum_{i=1}^{N} |Effort_i - Estimated\ Effort_i|\right) \quad (11)$$

The program parameter setting is modulated as shown in Table 1. The value of maximum iterations is ordered to 100, size of the population (or size of swarm) is ordered to 50, the Coefficient of Acceleration is set to 2.0 and 2.0, Coefficient of Inertia is set to 1 and 0.99, and Maximum velocity is set to 100 and the minimum velocity is to -10.

Table 1. Setting in Parameters of PSO

Parameter	Value
Maximum iterations	100
Number of Particles (Population Size)	50
Acceleration constant	[2.0, 2.0]
Inertia weight	[1,0.99]
Maximum velocity (or Vmax)	100
Minimum velocity (or Vmin)	-10

The experiment of applying the technique of PSO for optimizing coefficient of COCOMO II model using the dataset of Turkish Software Industry which consists of twelve data instances. Each data instance has of twenty-five attributes consisting of Project ID, five Scale Factor, seventeen Effort Multiplier within the range of value intervals from VeryLow to ExtraHigh. Size of a project is stated in kilo (thousands) of lines of code (KLOC) and Measurable Measures as an actual effort. Details of the dataset are shown in Table 2. All data are applied for calibrating. The results of calibration can be implemented for subsequent projects which they have similar properties.

Table 2. Data Set

Project No.	Size (KLOC)	Measured Effort	Effort Multiplier	Scale Factor
1	3.0000	1.2000	0.3508	19.9200
2	2.0000	2.0000	0.4538	18.8300
3	4.2500	4.5000	0.6473	18.6800
4	10.0000	3.0000	1.1213	10.3100
5	15.0000	4.0000	1.0841	19.2800
6	40.5300	22.0000	0.2379	8.4100
7	4.0500	2.0000	0.1965	7.4200
8	31.8450	5.0000	1.0837	19.7300
9	114.2800	18.0000	0.3734	27.2300
10	23.1060	4.0000	0.6500	20.8200
11	1.3690	1.0000	0.2250	15.3600
12	1.6110	2.1000	0.4109	19.1100

4. Results and Discussion

This section shows the results of experiments that have been achieved by applying the method proposed for the dataset. The purpose of this optimization is reducing the uncertainty of the coefficients in the COCOMO II model. Parameters A and B were obtained applying the PSO technique and afterward a comparison of the results obtained with the normal values of the coefficients and the coefficient values of the Tabu Search method [14].

Implementation of this method using Matlab, source code modified from the source code provided by Yang [23]. PSO is applied to the Turkish Software Industry dataset. The offered experiments apply the PSO technique for optimization using equation (1) and (2). The implementation of PSO in updating the velocity and position of the optimization using equations (5) and (6) for the parameter of A and equation (7) and (8) for the parameter of B. The calculated the parameters of both A and B can significantly make accurate of software project estimates. In Figure 2, it is shown that the PSO convergence process after each iteration is performed. Population sizes of 10, 20, 30 and 40 are explored to see process performance. We found that PSO convergence in all experiments with the same minimum error (eg MMRE equals 34.1939).

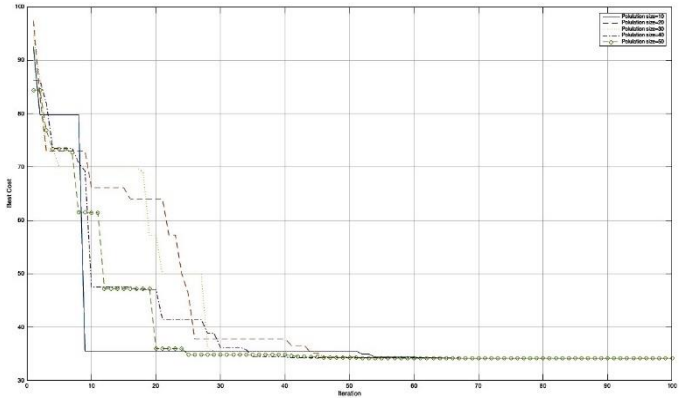

Figure 2. The best costs of MMRE using different size of the population which is 10, 20, 30, 40, and 50

After several iterations, we obtained the optimized results as A=4.3905 and B=-0.1829, not like default values in COCOMO II that A to 2.94 and B to 0.91. The new coefficient values are optimized, the calculated values of the effort are displayed in Table 3. While in Figure 3, the effort comparing for COCOMO II with the present in the graph shows that the calculated effort using PSO is much smoother and closer to the actual effort when it is compared with the effort predicted by simple values of coefficients and Tabu Search.

Table 3. Experimentation Result

Project No.	KLOC	Actual Effort	Effort of COCOMO II	Effort of Tabu Search	Effort of PSO
1	3.0000	1.2000	3.4881	1.9316	1.5679
2	2.0000	2.0000	2.8568	1.8909	2.0001
3	4.2500	4.5000	9.3041	4.4202	2.8581
4	10.0000	3.0000	33.9773	11.0776	4.0969
5	15.0000	4.0000	63.1555	17.2261	4.8891
6	40.5300	22.0000	27.7316	4.8844	0.7244
7	4.0500	2.0000	2.2887	1.1106	0.7411
8	31.8450	5.0000	147.0897	28.8072	5.0012
9	114.2800	18.0000	297.6050	33.2193	2.5041
10	23.1060	4.0000	63.9962	14.4335	3.0896
11	1.3690	1.0000	0.9239	0.7226	0.9789
12	1.6110	2.1000	2.0424	1.4868	1.8112

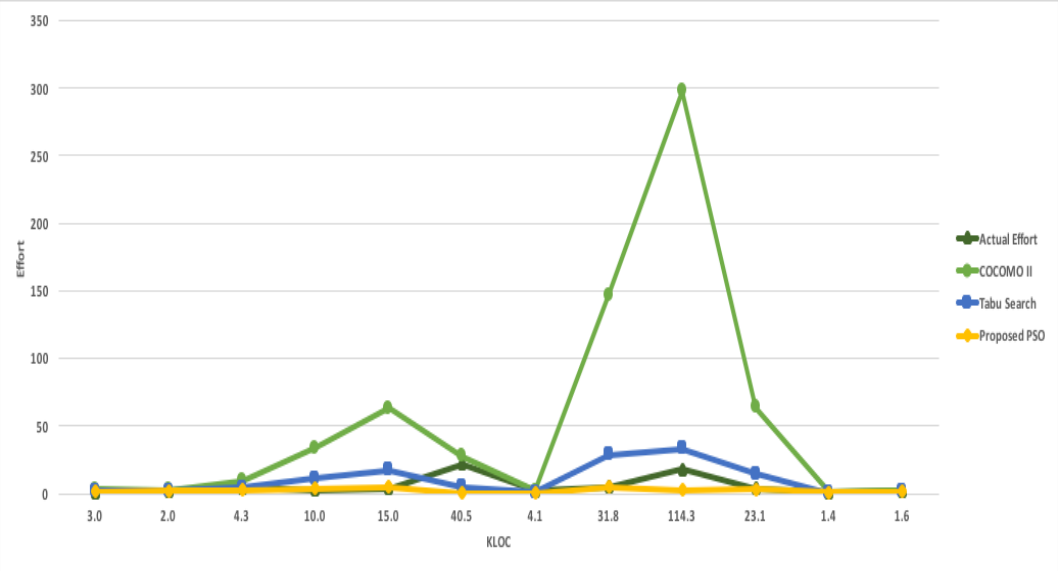

Figure 3. Graph of effort for the actual effort, the effort estimated using COCOMO II, the effort estimated using Tabu Search, and effort estimated using PSO

The function of fitness in equation (9), (10) and (11) is used for the calculation of experimental accuracy. The values are much better in terms of MMRE and MD in an attempt to estimate the actual effort. In Project Number of 2, 8, and 11, actual efforts are 2, 5, and 1 with the method offered. We can obtain accurate values as 2.0001; 5.0012; and 0.9789. This means that methods capable of minimizing errors are significantly down to 0.00%, 0.02%, and 2.11% to the actual effort.

As shown in Table 3, Project Number 1 has 3.49%, 1.93%, and 1.56% of MRE with calculated using the model of COCOMO II, Tabu Search, and offered PSO. The method offered can degrade 1.56%, 0.36% of MRE of COCOMO II and Tabu Search and so on with other projects. It gives results closer and more appropriate to the actual effort.

Such as previously discussed that MMRE states the accuracy of measurement, in this research MMRE value from experiment to 3 models is as follows: MMRE of COCOMO II is

733.1400%, MMRE of Tabu Search is 139.0699%, and MMRE of PSO is 34.1939%. This means that the PSO can degrade errors down to 698.9461% of the perspective of COCOMO II and 104.876% of Tabu Search perspective. Similarly above, MD of COCOMO II is 585.9266%, MD of Tabu Search is 90.5797%, and MD of PSO is 43.2477%. This means too that the PSO can reduce errors up to 542.6984% of the view of COCOMO II and 47.3320% of Tabu Search view as shown in Table 4 or Figure 4. The results of MMRE and MD indicate that the effort estimated by the method proposed gives better results than the COCOMO II and Tabu search.

Table 4. MMRE and MD for the Various Input Model

Model of Input	Model of Output	MMRE (%)	MD (%)
COCOMO II	For Effort	733.1400	585.9266
Tabu Search	For Effort	139.0699	90.5797
PSO Proposed	For Effort	34.1939	43.2477

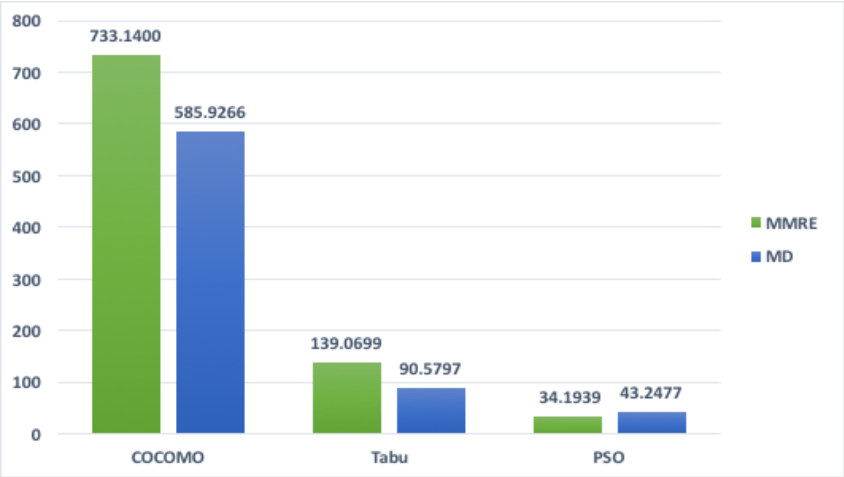

Figure 4. Comparison of MMRE and MD (in %) for COCOMO II, Tabu Search, and PSO

5. Conclusion

An accurate effort and cost estimation of software projects have been a challenge both for the software industries and the academic communities. A more accurate cost estimated of software projects can control more and more effective and efficient development resources. There are many models of software estimation which can be applied to estimate software development costs. In this paper, it was studied the efficiency of PSO implementation as an environmentally inspired technique of optimization algorithm to increase the precision of COCOMO II. The proposed PSO method is used to optimize the parameters using the dataset of Turkish Software Industry as test data.

The proposed method was assessed using the evaluation criteria. If it used MMRE as evaluation criteria, then the results showed that the PSO model could reduce to 698.9461% compared with regular COCOMO II model. The PSO model could reduce to 104.876% compared using the tabu search method. If it used MD as evaluation criteria, then the PSO could reduce to 542.6984% compared with the model of regular COCOMO II and the PSO could too reduce to 47.3320% compared to tabu search model. Therefore, the model of COCOMO II with optimized parameters of the PSO method provides a better estimate than the original COCOMO II model.

References

[1] Sachan RK, Nigam A, Singh A, Singh S, Choudhary M, Tiwari A, et al. *Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm*. Twelfth International Multi-Conference on Information Processing-2016 in Procedia Computer Science, Bangalore. Elsevier. 2016; 89: 492–8.

[2] Shekhar S, Kumar U. Review of Various Software Cost Estimation Techniques. *International Journal of Computer Applications*. 2016; 141: 31–4.
[3] Zahra SB, Nazir M. A Review of Comparison among Software Estimation Techniques. *University Journal of Information & Communication Technology*. 2012; 5: 7.
[4] Li YF, Xie M, Goh TN. A Study of Project Selection and Feature Weighting for Analogy Based Software Cost Estimation. *Journal of Systems and Software*. 2009; 82: 241–52.
[5] Sarno R, Sidabutar J, Sarwosri. *Comparison of Different Neural Network Architectures for Software Cost Estimation*. 2015 International Conference on Computer, Control, Informatics and Its Applications. Bandung. IEEE. 2015: 68–73.
[6] Sarno R, Sidabutar J, Sarwosri. *Improving the Accuracy of COCOMO's Effort Estimation Based on Neural Networks and Fuzzy Logic Model*. 2015 International Conference on Information, Communication Technology and System (ICTS). Surabaya. IEEE. 2015: 197–202.
[7] Mittal A, Parkash K, Mittal H. Software Cost Estimation Using Fuzzy Logic. *ACM SIGSOFT Software Engineering Notes*. 2010; 35: 1–7.
[8] Putri RR, Sarno R, Siahaan D, Ahmadiyah AS, Rochimah S. Accuracy Improvement of the Estimations Effort in Constructive Cost Model II Based on Logic Model of Fuzzy. *American Scientific Publishers*. 2017; 23: 2478–80.
[9] Huang SJ, Chiu NH, Chen LW. Integration of the Grey Relational Analysis with Genetic Algorithm for Software Effort Estimation. *European Journal of Operational Research*. 2008; 188: 898–909.
[10] Parwita IMW, Sarno R, Puspaningrum A. *Optimization of COCOMO II Coefficients using Cuckoo Optimization Algorithm to Improve the Accuracy of Effort Estimation*. 2017 International Conference on Information & Communication Technology and System (ICTS). Surabaya. IEEE. 2017: 99–104.
[11] Mansor ZB, Kasirun ZM, Arshad NHH, Yahya S. *E-cost Estimation Using Expert Judgment and COCOMO II*. 2010 International Symposium on Information Technology, Kuala Lumpur. IEEE. 2010; 3: 1262–7.
[12] Boehm B. COCOMO II Model Definition Manual. California. The University of Southern California. 1997.
[13] Kumar A, Sinhal A, Verma B. A Novel Technique of Optimization for Software Metric Using PSO. *International Journal of Soft Computing and Software Engineering*. 2013; 3: 2251–7545.
[14] Hari C, Reddy P. A Fine Parameter Tuning for COCOMO 81 Software Effort Estimation Using Particle Swarm Optimization. *Journal of Software Engineering*. 2011; 5: 38–48.
[15] Parkash J. COCOMO II Model Parameter Optimization using PSO and Effort Estimation. *International Journal of Information Technology & Mechanical Engineering*. 2014; 1: 1–11.
[16] Dejaeger K, Verbeke W, Martens D, Baesens B. Data Mining Techniques for Software Effort Estimation: A Comparative Study. *IEEE Transactions on Software Engineering*. 2012; 38: 375–97.
[17] Rao GS, Krishna CVP, Rao KR. *Multi Objective Particle Swarm Optimization for Software Cost Estimation*. In: Satapathy SC, Avadhani PS, Udgata SK, Lakshminarayana S, editors. ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India. Cham: Springer International Publishing. 2014; 248: 125–32.
[18] Ghatasheh N, Faris H, Aljarah I, Al-Sayyed RMH. Optimizing Software Effort Estimation Models Using Firefly Algorithm. *Journal of Software Engineering and Applications*. 2015; 08: 133–42.
[19] Sheta A, Rine D, Ayesh A. *Development of Software Effort and Schedule Estimation Models Using Soft Computing Techniques*. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). Hong Kong. IEEE. 2008: 1283–9.
[20] Baiquni M, Sarno R, Sarwosri, Sholiq. *Improving the Accuracy of COCOMO II Using Fuzzy Logic and Local Calibration Method*. 2017 3rd International Conference on Science in Information Technology (ICSITech). Bandung. IEEE. 2017: 284–9.
[21] Boehm B. Software Cost Estimation with COCOMO II. New Jersey. USA. Prentice Hall. 2000.
[22] Eberhart R, Kennedy J. *A New Optimizer Using Particle Swarm Theory*. Sixth International Symposium on Micro Machine and Human Science, Nagoya. IEEE. 1995: 39–43.
[23] Yang XS. Nature-Inspired Optimization Algorithms. 1st edition. Amsterdam Boston Heidelberg London New York Oxford Paris San Diego San Francisco Singapore Sydney Tokyo. Elsevier. 2014.