Optimizing COCOMO II Parameters using Particle Swarm Method

Kholed Langsari, Riyanarto Sarno
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
langsari@ftu.ac.th, riyanarto@if.its.ac.id

***Abstract*—The estimation of software effort is an essential and crucial activity for the software development life cycle. It is a problem that often appears on the project of making a software. A poor estimate will result in a worse project management. Several software cost estimation models have been introduced to resolve this problem. Constructive Cost Model II (COCOMO II Model) is a most considerable and broadly used model in cost estimation. To estimate the cost of a software project, COCOMO II model uses cost drivers, scale factors and line of code. However, the model is still lacking in terms of accuracy. In this study, we investigate the influence of components and attributes to achieve new better accuracy improvement on COCOMO II model. We introduced the use of Particle Swarm Optimization (PSO) algorithm in optimizing the COCOMO II model parameters. The proposed method is applied on Turkish Software Industry dataset. The method achieves well result and deals proficient with inexplicit data input and further improve a reliability of the estimation method. The optimized MMRE result is 34.1939%. It can reduce 698.9461% and 104.876% errors from the basic COCOMO II model and Tabu Search coefficient significantly.**

Keywords—cocomo ii model; software effort estimation; particle swarm optimization; swarm intelligence; optimization.

I. INTRODUCTION

Software Development is a systematic approach to Software Engineering discipline in constructing and maintaining a software system. The software project manager is the person who responsible for every software development life cycle activities. The primary job of the software project manager is to ensure that the project is accomplished with the goal of "high quality software with low cost within a certain time and budget". Estimation of software cost is the major challenge in a software project development. The accuracy of the estimation is vital in guiding software companies to create good management for the software's development. Moreover, good management of software development can estimate the cost and resources of the software precisely. It is calculated in term of person-month and it can handle both overestimates and underestimates of software effort and cost. This accuracy is derived from some variables or cost drivers. Getting an accurate software cost estimation needs accurate prediction method.

Several cost estimation methods have been proposed and improved by many researchers in the last few decades. These methods are categorized into Expert judgment, Algorithmic method, and Analogy based method. Constructive Cost Model (COCOMO) is the most well-known among all the software estimation model and is widely used in calculating software cost. Currently, many issues have arisen regarding the applicability of these methods to solve software cost estimation. Heuristic techniques are used to overcome the limitation of these methods and improve their applicability [3].

Various heuristic optimization methods are used in optimization problems. These methods can be used to estimate software cost as well. They are Particle Swarm Optimization [2, 11, 12, 13, 16, 17, 18], Genetic algorithm [14], Firefly Algorithm [15], and many others.

This paper presents a utilization of applying Particle Swarm Optimization (PSO) as an optimization algorithm in optimizing the COCOMO II model parameters so that a more realistic and accurate effort can be estimated. The remainder of the paper is organized as follows: In Section II, there are the literature review, a brief introduction of COCOMO model, and the basic principle of the methods discussed in this paper. Section III describes related works that have been researched. Section IV explains the methodological steps of work used in this experimentation. Section V describes the evaluation criteria and the dataset. Section VI presents the experiment and results from the comparison. Finally, Section VII concludes the study and result experimentation.

II. LITERATURE REVIEW

A. Software Cost Estimation

Software project management requires reliable software cost estimation to predict the amount of effort and resources in creating a software. The accuracy of cost estimation is crucial in developing a software. Estimating at early stages can help manage planning, budgeting, and monitoring of the activities in a project. Because there is a limited amount of resource for a project, accurate software estimation can provide sufficient support for the decision-making process, and all the development processes can be maintained efficiently and effectively. However, the most difficult problem in estimating software cost is the uncertainty and the complexity of the data

978-1-5090-5864-8/17/$31.00 ©2017 IEEE

that badly affects the process of software development. However, there are some techniques and procedures to handle this issue. Both algorithmic method and non-algorithmic method can help to estimate software cost. The algorithmic method usually uses linear regression method and the collection of previous data in its prediction. The non-algorithmic method tries to construct rules that fit the data. These include analogy method [9], artificial neural network [10], fuzzy [11], and genetic algorithm. Cost estimation is usually measured in terms of effort. The effort is the amount time of one person to work for a definite period of time. In general, when more effort is put, the more expensive the cost will be.

B. Effort Estimation Models

Various software effort estimation methods have been proposed to help project managers estimate and a make correct decision in building a software system with excellent accuracy in estimation [5]. COCOMO has become one of the most valuable and broadly used model of effort estimation. COCOMO was published by Barry Boehm in 1981 [6]. Effort and Schedule estimation models are the two main models delivered in COCOMO for software management. The model was developed from a dataset consisting of 63 projects. Each project was divided into 16 variables. COCOMO divides cost driver into 3 aspects, which are Effort Multiplier (EM), Line of Code (LOC), and Scale Factors (SF). All these cost drivers will be calculated with an equation to produce the amount of effort needed in person-months (PM). In 2000, Barry Boehm [7] introduced the COCOMO II model, which provides more accuracy with several improvements in some cost drivers. There are various software attributes used in the Post-Architecture Model phase of the COCOMO II model. The model consists of 17 Effort Multipliers (EMs) which are grouped into four categories with 5 Scale Factors (SFs). Effort Estimation, as result of estimation and Project Size, is represented in line of code (LOC) or thousand line of code (KLOC).

In the COCOMO II model [4], the equation that is used to calculate the software development effort is given in (1) and Scales Factors Computation is defined by (2):

$$\text{Effort(PM)} = A \cdot \text{Size}^{E} \times \prod_{i=1}^{17} \text{EM}_i + \text{PM}_{\text{Auto}} \quad (1)$$

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

A and *B* are the multiplicative and exponential constant, and have values of 2.94 and 0.91, respectively. *Size* is the estimated size of a project in Kilo Source Lines of Code (KLOC). *E* defines the scaling exponent for effort, which is an exponential factor which has a record of accounts associated with economies or diseconomies of scale which is extended as the software project size increases. EM_i is the Effort Multiplier where i = 1 to 17 and SF_j is the Scale Factor where j = 1 to 5. There are two constants for schedule calculation. Multiplicative constant C is the schedule coefficient and has a value of 3.67. Exponential constant D is the scaling base-exponent for the schedule that has a value of 0.28.

The purpose of this paper is to optimize two variations of the COCOMO II model parameters; multiplicative constant A and exponential constant B, to better improve its effort estimation.

C. Particle Swarm Optimization (PSO) Algorithm

PSO is a swarm-intelligence-algorithm based on natural behavior. It was presented in the year 1995 by Kennedy and Eberhart [7]. Because of the simplicity, durability, and flexibility of the PSO algorithm, it has become one of most well-known and broadly used swarm-intelligence-based algorithms. It uses randomness with real number, along with local and global communication among the swarm particles [1].

The PSO algorithm is used to search the space of an objective function by adjusting the movement of individual objects called "particles". Each particle tries to move toward the position to the global best ($g^{(t)}$) and its personal best ($x^{(t)}_i$) according to each one's best experience. When a particle of the swarm finds a position that is better than any previously found position, it will update the position as the new current best for particle i. After several numbers of iteration or when the objective no longer moves, finding the global best can be done by looking at all current best solutions. Fig. 1 illustrates the movement of a particle.

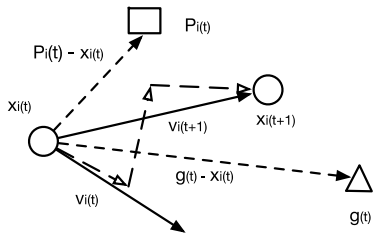

Fig. 1. Representation of how particle *i* moves to the current best. And the global best position.

Assume that a particle *i* has a vector position of x_{ij} and velocity of $v_{ij,}$. The formula for calculating new positions of the velocity vector is shown by the following:

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

The initial position for every particle should be shared with other particles reasonably so that they can easily stay in the group. The initial value of the velocity begins is 0 (zero), so now, $v^{t=0} = 0$. The particle's position can now be updated by the equation below.

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

The variable x_{ij}^{t} is the current position of particle *i*, x_{ij}^{t+1} is the new position of particle *i*, v_{ij}^{t} is the current velocity, v_{ij}^{t+1} is the moving velocity, $P_{best,i}^{t}$ is the personal best experience of the particle, G_{best}^{t} is the global best value, w is the weighting function, and r_1 and r_2 are two positional random vectors and generate values between 0 and 1. The acceleration variables c_1 and c_2 are the personal acceleration and acceleration coefficient parameters, and can approximately be set to 2 for both. PSO solution space ranges within [-x, x]. Although ***vi*** can be set to any possible solution values, it depends on the lower bound $[0, v_{min}]$ and upper bound $[0, v_{max}]$ of the decision variable range.

III. Related Work

There are various previous works that attempted to optimize and improve the accuracy and calibrate the parameter's value of COCOMO.

Riyanarto and Johannes [9] [10] investigated the roles of Effort Multiplier (EM) and Line of Code (LOC) to utilize effort estimation. Gaussian Membership Function (GMF) [9] has been applied to COCOMO II to represent the EM. GMF could create a smoother transition, resulting a more accurate Effort Multipliers. And they also applied Neural Network (NN) approach [10]. The proposed model shows a major improvement, compared to pure Fuzzy model or basic COCOMO model. Baiquni and Riyanarto [11] proposed a model based on Fuzzy Logic, Local Calibration, and Tabu Search. They tried to improve accuracy by fuzzifying cost drivers in Fuzzy Logic with Gaussian Membership Functions (GMF) to redesign the Effort Multiplier. Local Calibration as Calico and Tabu Search is used to find the value of parameters and give the new value for the calculation parameters of the COCOMO II model. The new value is able to improve accuracy and decrease error significantly.

Prasad Reddy et al. [17] and Ruchi Puri [18] proposed Multi-Objective Particle Swarm Optimization (MOPSO) model for software cost estimation. Jai and Kamabir [13] proposed PSO technique to optimize COCOMO II model coefficients with NASA dataset. They found that PSO has efficiently solved the optimization problem, reduced the uncertainties, and gave better results when compared to using regular value of coefficients. A. Kumar et al. [2] and Alaa Sheta [16] analyzed the optimization of PSO with linear regression and Fuzzy Logic. They implemented this on COCOMO model with NASA18 dataset. The PSO method gave an appropriate process in optimizing the prediction of the effort. The method of linear regression also generated fairly high results, but it's time-consuming during each process. Hari and Reddy [12] introduced an important generalization of the COCOMO and proposed the model by augmenting PSO with Constriction Factor in adjusting the constants. The experiment showed that the models deal well and capable while lacking accurate input, and is able to enhance the suitability for software effort prediction. Jai Parkash and Karambir [13] also proposed the use of PSO algorithm for optimization. They tried to optimize coefficients a, b, c, and d of the COCOMO II model. They did an experiment using Turkey Industry dataset with 15 data points. The results obtained using the PSO were better than the results obtained by using the regular value of coefficients. They performed optimization on four parameters, instead of effort estimation in Post Architectural Model which requires only parameter A and B.

IV. Methodology

There are various uncertainties in effort estimation using the basic COCOMO II model parameters and PSO technique. Our focus is on optimizing the multiplicative constant A and exponential constant B to calculate effort in Effort Estimation of COCOMO II model. In this paper, the methodology in optimizing the parameter of the COCOMO II model is optimized using PSO. Here, PSO acts as a global optimization technique. It is applied to investigate and resolve unpredictable input, optimize the parameters coefficient related to the effort, and produce a result in significantly less execution time.

PSO uses real number randomness and global communication among the swarm particles. The minimum fitness value and initial value to begin the optimization process is set. These two values are Personal Best (Pbest) and Global Best (Gbest). The processes of each iteration show the gaining of the best fitness value with certain improvements. In its iterations, each particle of swarm attempts to make changes and movements of its current position and current velocity depending on the length of space between its current position and the new Pbest position, as well as the length of space between current position and the new Gbest position.

The input is Software Project Size, Actual Effort, Effort Multiplier, and Scale factor. And the output of the process are the new values of the parameters A and B. The steps proposed are:

1. Giving "n" particles of the swarm random vector positions ($X_i^{(t)}$) and velocities ($V_i^{(t)}$) from optimizing parameters. The range between velocity [*Vmin, Vmax*] is identified as needed. Personal best (*Pbest*) for each particle is the initial position of each iteration.
2. Setting the standard weight function value (*w)* as 1, as well as weight for personal acceleration coefficient (c_1) and social acceleration coefficient (c_2) as 2.0.
3. Set the particles as *i* = 1 to n, set each particle position with parameters of optimizing value (see Table I), and evaluate the result in the fitness function. The function of fitness here is called the Mean Magnitude of Relative Error (MMRE) and Manhattan Distance (MD), (9), (10), and (11). The objective is to minimize the MMRE and MD by choosing an appropriate best result from the area of variation specified in step 1.
4. Evaluate and compare the value of *personal best (Pbest)* for every particle with *Effort* and *Actual Effort*, from current and previous values. If the new *fitness value (f)* is greater or better than *global best* (*Pbest*) fitness, then set (*Pbest) = new fitness value (f)*.
5. Set the best *Pbest* value to be the global best (*Gbest*). The best particle value is chosen by considering the

range of variations between *Effort* and *Estimated Effort* with the appropriate *Gbest* particle as an improvement step.

6. Update particle *i* position and velocity of the optimized parameter based on (3) and (4). The new regulation formula for updating position and velocity vector are in (5) and (6) for parameter A and in (7) and (8) for parameter B:

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

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

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

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

7. The *Gbest* values are the optimal solution value.
8. Repeat step 3 to 7 until the iterations number meet the user specified or particles state condition.

V. Evaluation Criteria and Data Set

We propose MMRE and the absolute difference between effort and estimated effort (MD) as the fitness functions for the proposed method.

The main objective of this estimation method is to verify whether the predictions are precise; the gap between the predicted of effort, $EstimatedEffort_i$, and the realistic actual effort, $Effort_i$ should be measured as close as possible. Large value difference between $Effort_i$ and $EstimatedEffort_i$ will reduce the accuracy of the prediction and can negatively affect the effort in the software system development. In this paper, Magnitude of Relative Error (MRE) [3] is applied as the common criteria on software cost estimation in evaluating the accuracy of the estimated effort. The MRE is considered to calculate each projected point as defined in (9):

$$MRE_i = \frac{|\text{Effort}_i - \text{Estimated Effort}_i|}{\text{Effort}_i} \times 100 \quad (9)$$

MMRE [3] is used to count the average value of the results from each individual accuracy prediction values that was measured in the MRE criteria, as shown in (10):

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

The Manhattan distance which calculates the absolute difference of distance between effort and estimated effort is calculated by (11):

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

The parameter settings of the program are set as in Table I. The maximum number of the Iterations is 100, Population Size or Swarm Size is set to 50, Personal Acceleration and Acceleration Coefficient are 2.0 and 2.0, Inertia Coefficient is 1 and 0.99, Maximum Velocity is 100 and Minimum velocity is -10. The parameters for method comparison as Tabu can be found in [11].

TABLE I. PSO Parameter Setting

Operator	Criteria
Iterations	100
Population Size	50
Weight Acceleration coefficient	[2.0, 2.0]
Weight Inertia coefficient	[1,0.99]
Maximum velocity (Vmax)	100
Minimum velocity (Vmin)	-10

The experiments apply PSO to optimize the COCOMO II model parameters based on the latest dataset from Turkish Software Industry. The dataset is obtained from five different software companies in various domains. It consists of data from 12 projects, with each project consisting of 25 attributes which includes Project ID, 5 Scale Factor, 17 Effort Multiplier in the value interval range from VeryLow to ExtraHigh, Measured Effort as actual effort, and Project Size represented in KLOC. The dataset details are shown in Table II. All project data points will be used in calibration. The results from the calibration can be used for future projects from a similar category.

TABLE II. Software Industry DataSet

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

VI. Experimentations And Result

This section presents the experiment and the results obtained by implementing the proposed method to the dataset. The main objective of the optimization is to reduce uncertain parameters from A and B coefficients of a COCOMO II model using PSO technique and compare newly obtained results with the general and Tabu Search [11] coefficient.

The method is implemented in Matlab. The source code is a modified and further-developed version from the one provided by X.-S. Yang [1]. PSO has performed on the datasets from Turkish Software Industry. The proposed experiment applies PSO for optimizing, as shown in (1) and (2). The implementation of PSO to update the positions and the velocity vectors of the optimized procedure is given in (5) and (6) for parameter A and in (7) and (8) for parameter B. The computed parameters can significantly simplify the estimation of software development effort for all projects.

In Fig. 4, we illustrate the confluence process of PSO after several iterations with different population sizes. Population size of 10, 20, 30 and 40 are inspected to perceive the performance of the processes. It was discovered that PSO confluence in every experiment has the same minimum error. After several iterations, we were able to obtain the newly optimized coefficient results with A=4.3905 and B=-0.1829, instead of the basic COCOMO II values which are A=2.94 and B=0.91. The results computed with the newly optimized coefficient values of effort are presented in Table III. In Fig. 3, the effort graph also shows that using PSO is much smoother and closer to actual effort when compared to efforts estimated by basic coefficients of COCOMO II and Tabu Search.

TABLE III. Computed Mean MRE (MMRE) and Manhattan Distance (MD).

Model Input	Model Output	Mean MRE (MMRE)	Manhattan Distance (MD)
COCOMO II Model	Effort	733.1400	585.9266
Tabu Search	Effort	139.0699	90.5797
Proposed PSO	Effort	34.1939	43.2477

The fitness function as shown in (9), (10) and (11) are used to calculate the accuracy of the experiment. In Table IV, Project No. 2, 8, and 11 has respective actual efforts of 2, 5, and 1. When using the proposed method, we were able to gain the accurate value as 2.0001, 5.0012 and 2.11. It means that the method is able to minimize error significantly, down to 0.00%, 0.02%, and 2.11% respectively. As shown in Table IV, Project No. 1 has achieved 3.4881%, 1.9316%, 1.5565% MRE values by using COCOMO II model, Tabu Search, and proposed PSO coefficients respectively. The proposed method can reduce 1.5565% MRE error from the COCOMO II model, and 0.3637% from the Tabu Search. The proposed method provides result much closer and more suitable to the actual effort.

The MMRE of each method represents overall of accurate measurement. The MMRE value of COCOMO II model, Tabu Search, and PSO as 733.1400%, 139.0699%, and 34.1939%. It is means that the PSO is able to reduce error up to 698.9461% from the COCOMO II model and 104.876% from Tabu Search. The result of MMRE and MD show that effort estimation by the proposed method delivered a much better solution when compared with COCOMO II model and Tabu search as illustrated in Fig. 2.

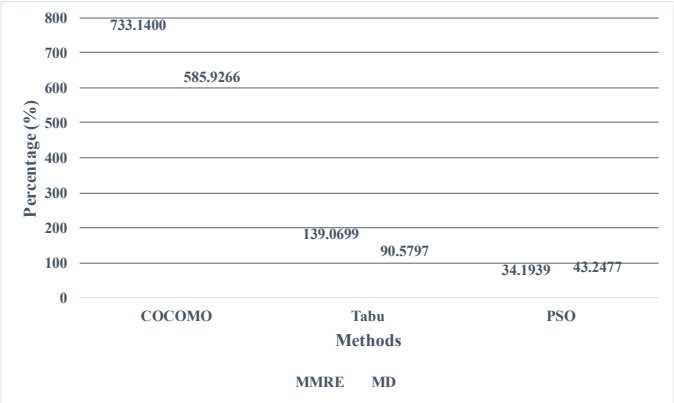

Fig. 2. Comparison of Mean Magnitude of Relative Error and Manhattan distance by COCOMO II, Tabu Search and PSO in percentage.

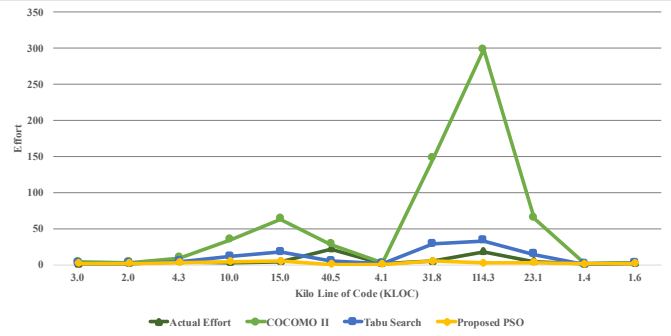

Fig. 3. Effort Graph for Actual Effort, COCOMO II, Tabu and PSO.

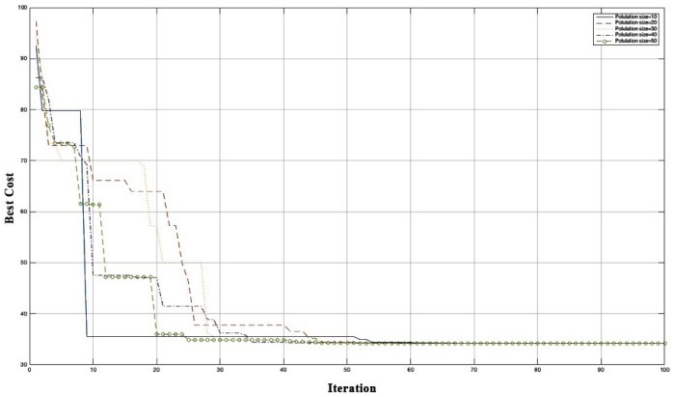

Fig. 4. Best cost of MMRE with various population sizes, population size 10, 20 ,30, 40, and 50

TABLE IV. EXPERIMENTATION RESULT

Project No.	KLOC	Actual Effort	COCOMO II Effort	Tabu Search Effort	PSO Effort
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

VII. CONCLUSION

The challenge in creating a reliable, trustworthy, and accurate software cost estimation has been studied and improved both in the software industry and academic field. A more accurate software cost estimation can handle software development resources more efficiently. Several software cost estimation models that applicable to applied for forecast software cost. In this paper, we investigated the efficiency of applying the swarm intelligence, PSO as an optimization algorithm to improve the accuracy degree of COCOMO II model by optimizing its parameters. The proposed PSO method has been implemented with the Turkish Software Industry dataset. The method has been assessed according to evaluation criteria. Evaluation results have shown that PSO is 698.9461% MMRE and 495.3469% MD lower than general COCOMO II model coefficient parameters, and 104.876% MMRE and 47.332% MD lower than Tabu Search. Optimizing parameters of COCOMO II model with PSO method gives an improved estimation compared to the basic COCOMO II model.

REFERENCES

[1] X.-S. Yang, *Nature-Inspired Optimization Algorithms*, 1st ed. Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V., 2014.

[2] A. Kumar, A. Sinhal, and B. Verma, "A Novel Technique of Optimization for Software Metric Using PSO," *International Journal of Soft Computing And Software Engineering*, vol. 3, no. 10, pp. 2251–7545, 2013.

[3] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: A comparative study," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375–397, 2012.

[4] B. Boehm, C. Abts, B. Clark, and S. Devnani-Chulani, *COCOMO II Model Definition Manual*. University of Southern California, Los Angeles, 1997.

[5] B. Boehm, Cost Models for Future Software Life Cycle Process: COCOMO 2, *Annals of Software Engineering*, 1995.

[6] B. Boehm, Software Engineering Economics, New Jersey, USA: Prentice-Hall, Englewood Cliffs, 1994.

[7] B. Boehm, Software Cost Estimation with COCOMO II, New Jersey, USA: Prentice Hall, 2000.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, 1995.

[9] R. Sarno, J. Sidabutar, and Sarwosri, "Improving the accuracy of COCOMO's effort estimation based on neural networks and fuzzy logic model," *International Conference on Information & Communication Technology and Systems (ICTS),* 2015. doi: https://doi.org/10.1109/icts.2015.7379898

[10] R. Sarno, J. Sidabutar, and Sarwosri, "Comparison of different Neural Network architectures for software cost estimation," *International Conference on Computer, Control, Informatics and its Applications (IC3INA).* pp. 68-73, 2015. doi: 10.1109/IC3INA.2015.7377748

[11] M. Baiquni and R. Sarno, "Optimizing Constructive Cost Model (COCOMO) for increase accuraracy of Estimation Effort," *JURNAL TEKNIK ITS*, vol. 4, no. 1, pp. 1–6, 2015.

[12] CH.V.M.K. Hari and P.V.G.D. Prasad Reddy, "A Fine Parameter Tuning for COCOMO 81 Software Effort Estimation using Particle Swarm Optimization," *Journal of Software Engineering,* 5: 38-48. 2011

[13] Jai Parkash et al, "COCOMO II Model Parameter Optimization using PSO and Effort Estimation," *Int. Journal of Information Technology & Mechanical Engineering - IJITME*, Vol.1 Issue. 4, pg. 1-11, 2014.

[14] R.K. Sachan, A. Nigam, A. Singh, S. Singh, M. Choudhary, A. Tiwari, and D. S. Kushwaha, "Optimizing Basic COCOMO Model using Simplified Genetic Algorithm," *Procedia - Procedia Computer Science*, vol. 89, pp. 492–498, 2016.

[15] N. Ghatasheh, H. Faris, I. Aljarah, and R.M. H. Al-sayyed, "Optimizing Software Effort Estimation Models Using Firefly Algorithm," *Journal of Software Engineering and Applications*, no. March, pp. 133–142, 2015.

[16] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008.

[17] G.S. Rao, C.V.P. Krishna, and K.R. Rao, "Multi Objective Particle Swarm Optimization for Software Cost Estimation," *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I*, vol. 32, no. 3, pp. 125–132, 2014.

[18] R. Puri, "Novel Meta-Heuristic Algorithmic Approach for Software Cost Estimation," International Journal of Innovations in Engineering and Technology (IJIET), vol. 5, no. 2, pp. 456–463, 2015.