TELKOMNIKA, Vol.16, No.5, October 2018, pp.2199~2207
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v16i5.9698

Optimizing Time and Effort Parameters of COCOMO II Using Fuzzy Multi-objective Particle Swarm Optimization

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

Abstract

Estimating the efforts, costs, and schedules of software projects is a frequent challenge to software development projects. A bad estimation will result in bad management of a project. Various models of estimation have been defined to complete this estimate. The Constructive Cost Model II (COCOMO II) is one of the most famous models as a model for estimating efforts, costs, and schedules. To estimate the effort, cost, and schedule in project of software, the COCOMO II uses inputs: Effort Multiplier (EM), Scale Factor (SF), and Source Line of Code (SLOC). Evidently, this model is still lack in terms of accuracy rates in both efforts estimated and time of development. In this paper, we introduced to use Gaussian Membership Function (GMF) of Fuzzy Logic and Multi-Objective Particle Swarm Optimization (MOPSO) method to calibrate and optimize the parameters of COCOMO II. It is to achieve a new level of accuracy better on COCOMO II. The Nasa93 dataset is used to implement the method proposed. The experimental results of the method proposed have reduced the error downto 11.89% and 8.08% compared to the original COCOMO II. This method proposed has achieved better results than previous studies.

Keywords*: multi-objective PSO, software effort estimation, COCOMO II, fuzzy logic*

1. Introduction

The manager of the software development project is the person responsible for controlling the activities or activities of the software development, from the needs analysis to the software maintenance process [1]. To get high-quality software using the fewest resources in the development team is the primary task of a software project manager [2]. Therefore, appropriate estimates of resource requirements are required to plan project execution. An overestimated estimate of actual resource needs will result in a waste of resources, otherwise too underestimates tend to result in a lack of resources during execution.

Management of software project requires a reliable estimate of software costs to make the assessment of the amount of effort and resources required to complete the project. The accuracy of effort and cost estimated for developing software is significant. Estimating in early efforts and costs can help project managers to plan, to budget, and to monitor project activities. Due to the resources limited to a project, profer software estimates can supply adequate support for an efficient and effective decision-making process. However, a hard problem for estimating the cost of software is the presence of unpredictable obstacles and the intricacies of data that create adverse effects on the software development process.

One of the major challenges in software project management is the cost estimated for software projects. To direct companies of software in creating the right management to develop software is required a better level of accuracy of project cost estimated. In addition, a nice management of software project can predict effort, cost, and software resources appropriately. It is stated in person-months. This can manage both the overestimates and the underestimates for software efforts and the costs required to complete the project. It can also manage the application's quick configuration [3]. This level of accuracy comes from several influential variables or cost drivers. Thus, to obtain precise cost estimated of software requires a right prediction method.

Received April 27, 2018; Revised June 22, 2018; Accepted September 8, 2018

Many methods to estimate cost have been presented and refined by several researchers. However, the COCOMO method has become one of the most well-known methods of estimating the effort of the most popular software projects today [4]. Generally, problems arise by regarding the accuracy of the estimation results of this method for the cost estimated of the software. Several heuristic techniques have been used to overcome the limitations of this method to improve the level of accuracy [5]. Some heuristic optimization methods are used to solve this problem. These methods include: Particle Swarm Optimization [6-10], Genetic Algorithm [11,12], Firefly Algorithm [13], Cuckoo Optimization Algorithm [14], and many others.

In this paper, we undertook the study of the implementation of Fuzzy Logic and Multi-Objective Particle Swarm Optimization (MOPSO) to calibrate the parameters in COCOMO II to obtain optimal estimation results. The rest of this paper is structured as follows. In Section 2 we will briefly describe the related work that has been investigated for earlier researchers to estimate efforts through different methods and through the PSO approach. In Section 3, we describe the steps of the working methodology used in this study. In Section 4, we present the results achieved and analysis of the results achieved. Section 5 is the last section, we conclude that the accuracy of estimates of effort can be improved through model and effort estimates.

2. Related Work

Previous studies have been conducted to calibrate the COCOMO II coefficient values intended to optimize and improve the accuracy of effort and cost estimation. Sarno and Sidabutar undertook to investigate the role of software measures stated in SLOC and Effort Multiplier (EM) on increasing the accuracy of the effort estimates [15]. In this study, Fuzzy Logic using Gaussian Membership Function (GMF) is practiced to EM of COCOMO II. GMF succeeded in making a smoother transition than previously, meaningful more precise EM. In addition, Sarno et al [16] also used the trapezoid membership function to improve the precision of COCOMO II. Similar to previous research, fuzzy logic is applied in EM from COCOMO II.

The researchers also applied Neural Network (NN) as an approach using a multi-layer feed-forward of the neural network with a learning algorithm of back-propagation [17]. The model can improve upon the basic model of fuzzy or the original COCOMO. Baiquni and Sarno promoted a model that was an integration between Fuzzy Logic and Tabu Search to perform local calibrations [18]. These researchers have increased precision by obscuring cost drivers in Fuzzy Logic using GMF to reconstruct the EM. Local calibration such as Tabu Search and Calico are applied to discover new parameter values for COCOMO II calculations. The value of the new coefficient on COCOMO II can increase accuracy and reduce errors significantly.

Parkas and Kamabir [8] have used PSO techniques to optimize coefficients of COCOMO II models with NASA datasets as test data. These researchers found that PSOs could solve optimization problems and reduced uncertainties and delivered better results than those obtained using the original coefficient values in COCOMO II. Kumar et.al [6] and Sheta et.al [13] analyzed to use PSO as the optimization with both Linear and Fuzzy Logic regression by constructing a set of linear models to reduce errors of uncertain costs. These researchers have emphasized the COCOMO II model using the NASA dataset. PSO provides an efficient way to optimize business and cost predictions, while linear regression methods deliver great results but take a long time. Reddy et.al [7] promoted a prominent generalization for COCOMO II and promoted two models by adding PSO with factor of constriction for fine-tuning parameters. This model efficiently handles improper and uncertain inputs and improves the reliability of software estimates. The results of these experiments showed that PSO with a narrowing factor always leads to satisfactory results. Reddy et al [7] proposed a model for the estimated cost of software using Multi-Objective Particle Swarm Optimization (MOPSO). They inspected that the model gave better results when it was contrasted to the original COCOMO model.

2.1. Cost Constructive Model (COCOMO) II

Many models estimation in software cost have been proposed by several researchers to provide accurate and high-quality estimation results to help managers of software projects to make informed decisions regarding projects which they handle [19]. One of the most well-known and widely used estimation models is the Constructive Cost Model (COCOMO). COCOMO was published by Barry Boehm in 1981 [20]. The COCOMO is used as an estimate of cost models,

efforts, and schedules for planning software development activities. This model was built from a data set consisting of 63 data points which they have sixteen variables. In COCOMO, cost drivers were categorized into three aspects that is Effort Multiplier (EM), Source Line of Code (SLOC), and Scale Factors (SF). Every cost drivers will be determined by the equation (1) producing the effort in person-month (PM). Barry Boehm, in 2000, presented the more accurately provided model of COCOMO II with several aspects to improve some cost drivers.

If we use the post architecture of COCOMO II model than several attributes affect effort and cost estimates include: seventeen Effort Multipliers (EM), five Scale Factors (SF), and software sizes stated in Kilo SLOC (KSLOC). The equation is used to obtain the estimation effort given in equation (1).

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

The description of equation (1) is as follows: A is a constant that has a value of 2.94 as the default value if historical data does not exist. Size is defined as the size of software estimated in KSLOC, E is the scale-exponent. E is an exponential factor which it has account records for relative economies or scale diseconomies encountered due to the size of software projects increased, EM_I is The multiplier effort where i = 1, 2, 3 ... .17 for the post-architecture model. The coefficient E is computed using five Scale Factor by the equation (2).

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

where, B is an exponential constant having a default value (if historical data does not exist) to 0.91 and SF_j is a Scale Factor where j = 1, 2 ... 5. While to obtain the time of development (TDEV) we used equation (3) and to obtain exponential effort F is obtained by equation (4).

$$TDEV = [C\ x\ (PM_{NS})^{F}] x \frac{SCED\%}{100} \quad (3)$$

$$F = D + 0.2 \times [E - B] \quad (4)$$

where C is a constant of development time having a default value of 3.67 and D is an exponential having a default value of 0.28. F stated the scale-exponent for the schedule of development time. In this paper, it is proposed to calibrate the constants A, B, C, and D to obtain optimal estimation results. Four variations of calculation of effort and schedule calculation parameters of COCOMO II for better improvement than original COCOMO II using Fuzzy Logic and MOPSO for the dataset of NASA.

2.2. Fuzzy Logic

Fuzzy logic (FL) was first introduced by Zadeh in 1965 [21], the term is given for mathematical systems which were constructed to model the way human intelligence reasoning when word processing. The main characteristic of FL is the lack of accuracy in the process of measurement. Zadeh states that when complexity increases, exact statements miss meaning, and statements miss a significant degree of precision [21]. Fuzzy logic provides a way that enables to handle both qualitative and quantitative data in a single model. It is a multi-value logical form raised from the theory of the fuzzy union to deal with more precise reasoning than a prediction. The fuzzy set is a set whom it's elements have membership degrees [22]. Some functions of membership in fuzzy logic may be in form Trapezoidal, Triangle, Gaussian, and others. In this research be applied GMF as membership function.

Fuzzy Logic System (FLS) is a given method for a system consisting of the relationship between fuzzy and principles of fuzzy logic. The most prominent FLS is categorized into three types that are basis FLS, Takagi and Sugeno's fuzzy systems, and FLS using fuzzifiers and defuzzifiers [23]. Most of the application techniques by making inputs use sharp data, then producing a sharp data output. The FLS using fuzzifier and defuzzifier is applied widely that the fuzzifier renders the sharp inputs to the fuzzy set, and then defuzzifier renders fuzzy sets to the sharp output. Figure 1 described the system of logic which is promoted by Mamdani [24].

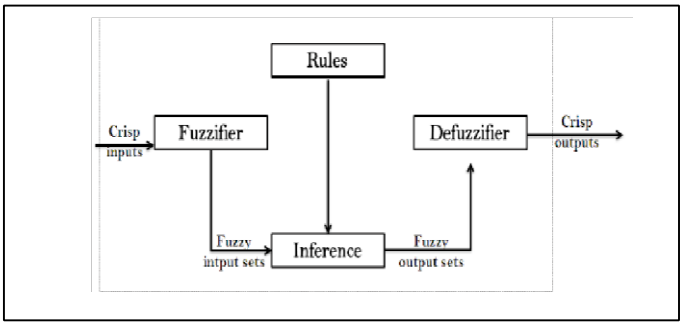

Figure 1. System of Fuzzy Logic using fuzzifier and defuzzifier application

2.3. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) was first familiarized in 1995 by Kennedy and Eberhart [25]. PSO is a swarm intelligence algorithm based on the inspirational behavior of swans. PSO has become one of the famous and widely used intelligence algorithms because it is based on the durability, simplicity, and flexibility of this algorithm. PSO employs randomness using real numbers, and local and global communications between the herd particles [24]. The PSO is set to get spaces of objective function updating the motion of individual objects named 'particles'. Every particle attempts to shift towards the best global position $g^{(t)}$ and its personal best $x_i^{(t)}$ conforming to the best experience. At the time when the herd particles found a better position than the position previously found, so the particle updates its position becoming the best new current position of particle i. Subsequent through some number of iterations or goals no longer moving and improving, so the best of all the best solutions have achieved. Let x_{ij} and v_{ij} be the vector of positions and velocities of particle i. The new vector of velocity is considered by equation (5).

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

The starting location of all particles must be uniformly distributed, so they can obtain the sample in most areas that are essential for multimodal problems. The starting velocity for a particle can be initialized as 0 (zero), which is, $v^{t=0} = 0$. Then, the new position can be renewed using equation (6).

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

Where, x_{ij}^{t} is the current search point; $x_{ij}^{t+1}$is a modified search point; x_{ij}^{t+1} is the current velocity; v_{ij}^{t+1} is the modified rate; $P_{best,i}^{t}$ is the best experience of any particles; G_{best}^{t} is the best in the world; *w* is a weighted function; r_1 and r_2 are two vectors at random, and each entry takes values between 0 and 1. Parameters c_1 and c_2 are learning parameters or acceleration constants, which can usually be taken as, say, c1 ≈ c2 ≈ 2. In the swarm optimization technique particles, looking for solutions in solution space in the range [-x, x]. Even though v_i can have any value, but it is normally limited in the range [0, v_{max}].

2.4. Multi-Objective Particle Swarm Optimization (MOPSO)

The basis of a single objective in problem of optimization is formulated for minimum or maximum such Minimum or Maximum $f(x)=[f_1(x), f_2(x), \ldots, f_M(x)]$, subject to $g_j(x) \leq 0$, j= 1, 2, …, J, and $h_k(x) = 0$, k=1, 2, …., K, a solution minimizes the scalar $f(x)$, where x= (x_1, x_2, …, x_d) is the vector of decision variables. In some formulations used in the optimization literature, inequalities g_j (j =1, …, J) can also include any equalities, because an equality $\emptyset(x)=0$ can be converted into two inequalities $\emptyset(x)\leq 0$ and $\emptyset(x)\geq 0$. However, for clarity, here we list the equalities and inequalities separately [24,26].

On a real issue, someone always involves optimizing with more than one goal. Multi-purpose optimization not always and should get a solution in optimal which can minimize

simultaneously all multi-objectives. The parameters of some goals in optimal generally do not cause to optimal other purposes [27]. Thus among these conflicting objectives, we choose just a few and ignore the other to achieve a certain balanced purpose. Next, we analyze by comparing the various objectives and then making compromises. Typically, we need to reformulate and find valuable scalar functions that represent a weighted composite or a preference sequence of all destinations [24].

Making or transferring to a single objective of PSO, each objective has its own weight, then we integrate those objectives into a formula of single weighted using equation (7) and normalize the weighted sum method applying equation (8).

$$F(x) = W_1 f_1(x) + W_2 f_2(x) + \ldots + W_M f_M(x) \quad (7)$$

$$\sum_{i=1}^{M} W_i = 1, w_i \in (0,1) \quad (8)$$

3. Research Method

There are various uncertainties of parameter values to determine the estimation of effort and time of software development using the original COCOMO II. In this study, we optimize the parameter values of the constants of multipliers and exponent that are A, B, C, and D on COCOMO II. The methodology to optimize the parameters of COCOMO II, in this study we use GMF of both Fuzzy Logic and MOPSO.

3.1. Fuzzy Logic

Here we describe step by step the implementation of the proposed approach of Fuzzy Logic. The use of Fuzzy Logic method referred to Sarno's research [17] and Sarno et al [18]. In this study, we learned the Effort Multiplier (EM) in the model of COCOMO II. Every EM applies linguistic magnitude representing the character of every EM. Driver cost which uses linguistic magnitude has among others: Very Low, Low, Nominal, High, Very High, and Extra High.

In this study, we categorize the EM into two groups that are quantitative and qualitative EM. The quantitative EM consists of DATA, RUSE, CPLX, TIME, DOCU, ACAP, PVOL, STOR, PCAP, APEX, PCON, LTEX, PLEX, SITE, SCED, and TOOL, while the rest includes quantitative EM. The Fuzzy model is applied for redesigning quantitative EM by reason of description of EM which can be transferred as Fuzzy Logic. As an instance, the effort multiplier: The LTEX has ranged from Very Low until Very High. The distinction of each degree is the percentage of using execution time available. In this research, we use GMF as the function of membership. GMF can create smoother transitions from one degree to another. The Fuzzy Logic is applied using the toolbox of fuzzy logic in MATLAB. This tool box is called the editor of Fuzzy Inference System (FIS). The GUI on FIS of the editor can help us making inputs and outputs with a number of membership functions which we require. With the editor of FIS, we compose rules ranging from input through output. The rules, in this research, are arranged as follows:

Rule 1: IF Input LTEX is low THEN Output data is increased
Rule 2: IF Input LTEX is nominal THEN Output data is unchanged
and so forth...

Figure 2 shows the Membership Function as an Input of one of the multipliers of effort ie LTEX. A description of LTEX for each level translated to GMF is also given. Suppose LTEX has descriptions of very low ratings given for less than 2 months experience, while very high ratings are awarded for 6 years or more experience, so we write low intervals to less than 2 and so on.

Figure 3 represents the function of membership of the Output for the effort multiplier: LTEX. The GMF value is set from the degree of each effort multiplier of LTEX. As an example, level 'Low' has a grade of 1.20, so we write down the interval down to 1.20. In the wake of creating the membership function for both input and output in GMF, we define rules, so assign new values to every degree created. Finally, we determine the same for the rest of the qualitative EM others. After obtaining a new EM value, we substitute the values in the dataset using the new value which resulted from Fuzzy COCOMO. Finally, from the results of this process, we get a new ranking table for calibration, then we use this information to optimize the parameters with MOPSO.

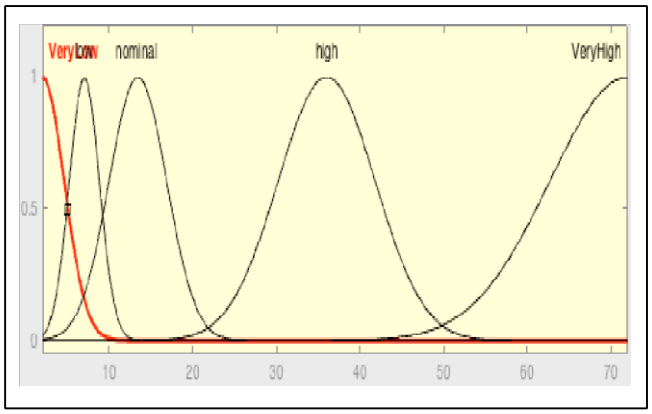

Figure 2. Representating Input of EM for LTEX using GMF

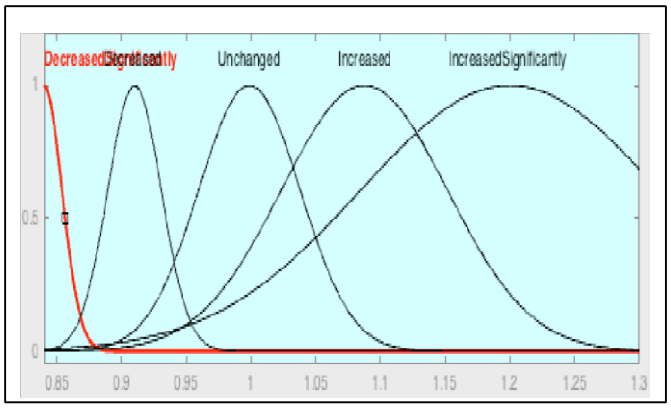

Figure 3. Representating Output of LTEX EM using GMF

3.2. Multi Objective Particle Swarm Optimization

The MOPSO as the promoted method is utilized accommodating Fuzzy of COCOMO efforts and estimated time of development. This approach proposed needs seventeen EM, five SF, actual effort, and TDEV. The MOPSO applies as a technique of global optimization. This applies by investigating and resolving unpredictable inputs and optimizing the coefficients of parameters associated with the effort and producing results in fewer implementation times

The steps of MOPSO to optimize parameters in COCOMO II as follows: Step 1: Initialization of m particles with random position and vector velocity [p_1, p_2… p_m] and [v_1, v_2…v_m] are suitable for the used parameters to be optimized. Step 2: Initiate each particle like P_{best} particle. Step 3: Assign fitness function $f_1(x)$, $f_2(x)$ putting on equations (1), (3), (9), and (10) for each particle. The purpose of $f_1(x)$ is minimizing and the goal of $f_2(x)$ is maximizing. Step 4: Change from Multi-Objective into Single-Objective with utilizing the method of the weighted sum.

Every one of the two purposes gives a rating for each particle. Next, enter the target range and assign it into each particle, so the last fitness is a minimized value. Step 5: If the particle fitness (p) is better than the Personal Best (P_{best}) fitness then P_{best} = Particle *(p)*, so set the best from P_{best} as Global Best (G_{best}). Step 6: Update both the velocity and position of a particle using Equations (5) and (6). So, step 7: Repeat step 4 through 8 until the particles do not change or move in the destination. While step 8: Give the parameter value of G_{best} as the optimal solution. Finally, the results provide the optimal value of optimization method. Then, parameter values are applied by calculating better new results for effort and time of development in COCOMO II.

3.3. Evaluation Criteria and Data Set

We use Mean Magnitude of Relative Error (MMRE) as the functions of fitness for the proposed method. The main question for each estimation method is whether the prediction is more accurate than before, the difference between the Actual Effort, the Effort, and the Effort, should be as small as possible. The major deviations between Effort i and Effort Estimated i will have a significant impact on costs associated with software development. In this paper, we use the most ordinary criteria in software cost estimates to evaluate the precision of the effort expected is MRE. MRE is obtained by calculating for each observation (each project) and defined as in equation (9).

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

Based on the MRE criteria, the number of measurements of accuracy is formulated. The MRE value of individual prediction can be averaged, resulting in Mean MRE (MMRE) [25] as a reward in equation (10).

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

The program parameter setting is set as showed in Table 1. This experiment implemented MOPSO to optimize parameters in COCOMO II based on the dataset of NASA93. The dataset has 93 projects as the data point. Each project is composed by 27 attributes that consist of Project ID, five SF, seventeen EM within the range of value intervals from 'Very Low' until 'Extra High', actual effort in man-month, Project size that be stated in SLOC, and the actual time of development in month. All project data would be applied to calibrate COCOMO II. The results of calibrating can be applied to subsequent projects of similar categories.

Table 1. Settings of MOPSO Parameter

Parameter	Value
Iterations	200
Population and Size of Repository	200, 100
Coefficient of Weight Acceleration	[1.0, 2.0]
Coefficient of Weight Inertia	[0.5, 0.99]
Maximum & Minimun Velocity (Vmax and Vmin)	10, -10
Minimum Velocity (Vmin)	-10
Rate of Inflation (alpha), Pressure of Leader Selection (beta), Pressure of Deletion Selection (gamma)	0.1, 2, 2
Rate of Mutation	0.1

4. Results and Analysis

In this part, we display the experimental results of the method promoted in implementation to the dataset. The major purpose of this optimization is to degrade the uncertainty in the coefficients of COCOMO II. The coefficient of parameters obtained using Fuzzy and MOPSO techniques was A, B, C, and D, so they were compared to the base coefficients in COCOMO II. Implementation of the method uses MATLAB, while the parameters calculated could significantly ease the effort estimates for all projects, after applying in several iterations. Subsequent to several iterations, it was obtained the results of the optimized new parameters A equals 4.3852, B equals 0.2830, C equals 2.7802, and D equals 0.3615 not the base value of COCOMO II which A equals 2.99, B equals 0.91, C equals 3.67, and D equals 0.28.

The result in the execution of the method proposed has been intended reducing MRE and MMRE error, so the smaller MRE or MMRE is near actual effort and actual TDEV. As an instance, ProjectID=9 (row 2 in Table 2) has 58.31% and 24.33% errors for effort and TDEV if using standard parameters of COCOMO II, while 53.05% and 13.31% error and if using the method proposed. The results of execution present that the method proposed is able to pull down 5.26% (58.31% minus 53.05%) and 11.02% (24.33% minus 13.31%) of the default setting in COCOMO II.

The MMRE of methods deputizes the MEAN's measurement precision. The average of MMRE by COCOMO II and Fuzzy MOPSO is 50.58% and 38.69% for effort estimated and 19,98% and 11,90% for TDEV estimated. It shows that the method promoted for reducing errors down to 11.89% and 8.08% from COCOMO II's perspective. The MMRE results also present that effort estimated and TDEV estimated with the method promoted provide a better estimation when contrasted to the original parameters of COCOMO II as shown in Figure 4.

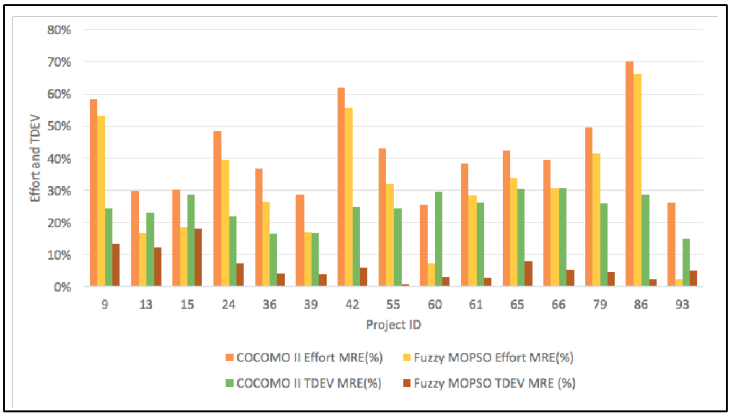

Figure 4. Comparing in Magnitude of Relative Error for the effort and the time of development between using original COCOMO II and Fuzzy MOPSO

Table 2. Comparing in MMRE for Effort Estimated and TDEV Estimated between using original COCOMO II and using Fuzzy MOPSO

ProjectID.	Effort by COCOMO II	Effort by Fuzzy MOPSO	TDEV by COCOMO II	TDEV by Fuzzy MOPSO
9	58.31	53.05	24.33	13.31
13	29.89	16.69	22.94	12.19
15	30.00	18.48	28.70	18.22
24	48.20	39.50	21.69	7.41
36	36.75	26.45	16.50	4.06
39	28.62	17.00	16.82	3.93
42	62.03	55.85	24.58	5.92
55	43.02	32.19	24.36	0.76
60	25.53	7.25	29.56	2.97
61	38.41	28.39	26.19	2.65
65	42.26	33.76	30.45	7.73
66	39.52	30.61	30.64	5.34
79	49.61	41.42	25.89	4.38
86	70.05	66.30	28.74	2.12
93	26.18	2.09	15.10	4.94
MMRE (%)	50.58	38.69	19.98	11.90

5. Conclusion

Research on the cost estimated of software is a challenge in a practical and academic level. An estimated cost, effort, and development time of a more accurate software project can handle resources more precisely, effectively, and efficiently. So far, there are more estimation models of software cost that can be applied to the cost of the software. In this study, we examined the efficiency of the application of the GMF of Fuzzy Logic and the MOPSO as calibrating and optimizing algorithm to refine the estimation results in the COCOMO II. The method proposed has been implemented with test data using the NASA93 dataset. The performance of this method is analyzed using MRE and MMRE as evaluation criteria. As it turns out, the method proposed could reduce MMRE significantly and the results of the evaluation show that calibrating and optimizing with the method proposed provides better estimation if it is compared with the original COCOMO II.

References

[1] Pressman R. Software Engineering: A Practitioner's Approach. Sixth Edition. Boston. Palgrave Macmillan. 2005.

[2] Kerzner, Harold, HR Kerzner. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. 11th Edition. New Jersey. USA. John Wiley & Sons. 2013.

[3] 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.

[4] 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.

[5] Boehm B. COCOMO II Model Definition Manual. California. The University of Southern California. 1997.

[6] 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.

[7] 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.

[8] Parkash J. COCOMO II Model Parameter Optimization using PSO and Effort Estimation. *International Journal of Information Technology & Mechanical Engineering*. 2014; 1: 1–11.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] Boehm B, Clark B, Horowitz E, Westland C, Madachy R, Selby R. Cost models for future software life cycle processes: COCOMO 2.0. *Annals of Software Engineering*. 1995; 1: 57–94.

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

[21] Zadeh. Fuzzy Sets. *Information and Control*. 1965; 8: 338--353.

[22] Muzaffar Z, Ahmed MA. Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems. *Information and Software Technology*. 2010; 52: 92–109.

[23] Wang L. Adaptive Fuzzy System and Control: Design and Stability Analysis. New Jersey. Prentice-Hall. 1994.

[24] 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.

[25] 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.

[26] Kumar V. Multi-Objective Particle Swarm Optimization: An Introduction. *The Smart Computing Review*. 2014; 4.

[27] Coello CCA, Reyes SM. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. *International Journal of Computational Intelligence Research*. 2006; 2.