Implementation of Bat Algorithm for COCOMO II Optimization

Yutika Amelia Effendi, Riyanarto Sarno, Joko Prasetyo
Department of Informatics, Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
e-mail: yutika.effendi@gmail.com, riyanarto@if.its.ac.id, joko.prsty@gmail.com

Abstract—**Software effort estimation is one of the main activities in software development project, because it can indicate whether the project will be successful or failed depending on the accuracy of the estimation. There are some models to estimate effort for software development. One of the well-known models is COCOMO II. The accuracy of COCOMO II effort estimation can still be improved by optimizing its constants. Several research have been conducted to improve COCOMO II constants, such as Differential Evolution, Harmony Search and Genetic Algorithm. In this paper, a model to optimize COCOMO II constants is proposed. The optimized constants are constant A and constant B by using Bat Algorithm with Turkish project dataset. The results of experiment are compared with local calibration and COCOMO II original constants by evaluating the proposed method using Mean Magnitude Relative Error (MMRE). The results show that the proposed model has the smallest value, which means its effort estimation has closer value of actual effort than that of the other methods which exist in the Turkish project dataset.**

Keywords—software effort estimation; COCOMO II; bat algorithm; MMRE; turkish project dataset; software project

I. Introduction

Nowadays, development of software has become one of the most important thing to do for organizations. Software which has been used for long time needs to be updated and upgraded regularly to maintenance it performance. Therefore, software project estimation is gaining attention in software project management because of it importance. In the implementation, software project estimation includes several components, namely risk analysis, cost estimation, quality estimation and effort estimation [1]. When software management decisions are being made, powerful assistance of software project estimation in term of accuracy is the main key. The accuracy of effort estimation can greatly help organizations to analyze the feasibility of a project, to manage the software development process in order to reduce risks [2].

Meanwhile, inaccuracy of effort estimation can affect the software quality and the commercial reliability of software companies in a negative way [3]. On the other hand, high estimation may cause some risks for an organization, such as loss of customer and unable to compete in a competitive market place. Hence, the accuracy of effort estimation, especially in the early stages of projects needs to be improved because it is always believed to be an important goal in the software industry. For more information, the Director of OUSD, department of System Engineering and Evaluation, DR. Patricia Sanders, addressed in one of her conferences in software technology in 1998 that the organizations spent cost about $40 in software development just to rework the software. In additional, the organization went through annual loss of $18 billion in 2000. Moreover, she stated that only 16% out of 100% of software developments could be finished on time and within budget.

In fact, estimating the effort for software is difficult because of the intangible nature of the software itself. Besides that, the software usually tends to change during its development. This changes may require unexpected modification in the estimation. In last few decades, software industry has been introduced some models which can help to estimate software effort. One of the most famous software cost estimation model is Constructive Cost Model (COCOMO). COCOMO was researched and published by Barry Boehm in 1981 [4]. The model was observed from dataset which consist of 63 data points and 3 aspects of cost driver. Each data point was divided into 16 variables. Cost driver divided into Effort Multiplier (EM), Scale Factors (SF) and Line of Code (LOC) which will be used to calculate the number of effort in person-months (PM) using some formulas. Barry Boehm also proved COCOMO II with some improvements in several cost drivers in 2000 and the result was more accurate than his research before [5].

A lot of research were conducted to optimize effort estimation of COCOMO. The methods which are used in COCOMO are Differential Evolution [6], Fuzzy [7 , 8], Harmony Search [9], Water Drop [10], Genetic Algorithm [11], Tabu Search and Genetic Algorithm [12]. Meanwhile, there are algorithms, such as Bee Colony Optimization and Firefly Algorithm which are used to optimize the constants of COCOMO II [13]. In COCOMO II, there are several constants which affect the accuracy of estimation. Those constants are A, B, C and D. To get optimal constants, calibration method can be applied to local projects which has been done beforehand. Therefore, those constants can also be optimized using other methods. Metaheuristic algorithm is one of the methods which can be optimized those constants.

According to a lot of research that have been done before, the results show that the constants A and B of COCOMO II can still be improved and Metaheuristic algorithm is the algorithm which can be implemented to improve the

COCOMO II constants, because it based on population and has some equations to avoid local optimum using global search and converge to the solution using local search. In this paper, we propose a model to optimize COCOMO II constants A and B using metaheuristic search bat algorithm.

II. RELATED WORK

A lot of research have been conducted before to optimize COCOMO model. In [14], NASA dataset that contains 60 projects were processed using Harmony Search to determine the similarity measures factor of software effort and to reduce the error of MMRE. In addition, NASA software project dataset implemented the Differential Evolution method. This method was helpful to estimate the COCOMO model parameters or constants [6]. Beside that, Neural Network and Fuzzy Logic are some algorithms to optimize COCOMO model [7, 8]. The multi-layer Neural Network which works feed-forward with back-propagation are usually implemented to solve the case. Sigmoidal activation function is applied only in hidden layer while output layer uses linear function. This model was also applied in NASA dataset.

In additional, to optimize the parameters of COCOMO model, we can use Simplified Genetic Algorithm (SGA) [11]. The basic idea of SGA was also implemented on NASA software project dataset. After the SGA was implemented, the experimental results showed that this algorithm had better estimation approaching the reality over basic COCOMO. For optimizing the COCOMO II, Genetic Algorithm can also be implemented by doing post architecture model by tuning its constants to improve the accuracy of predicting both the effort and the development time. Experiments were conducted to compare the accuracy of the constants with current COCOMO II constants on NASA data set.

Intelligent Water Drops algorithm is also used to optimize the coefficients of COCOMO II model. This algorithm generates more accurate estimation in terms of effort and time for software development. This algorithm has been used in NASA 93 software project dataset [10].

Next, there are Bee Colony Optimization (BCO) algorithm and Firefly algorithm [13]. BCO algorithm were compared to Halstead model, Bailey-Basil, COCOMO II and Walston-Felix. The basic idea of BCO algorithm approach is the algorithm can generate various partial solutions and get the best solution based on Mean Magnitude of Relative Error (MMRE). This concepts have been tried in experiment and the results show that BCO algorithm is able to improve the accuracy of cost estimation and get better solution than other models. Meanwhile, Firefly algorithm is able to minimize error value significantly. Not only minimize the error value, but also show the high accuracy compared with other meta-heuristic optimization algorithms, namely Particle Swarm Optimization (PSO) [15] and Genetic Algorithms (GA). Teaching Learning Based was used in NASA 93 projects dataset to optimize constants of COCOMO II and showed increasing efficiency.

A lot of research have been conducted before to optimize COCOMO II model. Last, there is Tabu Search and Genetic Algorithm. Tabu Search algorithm, one kind of meta-heuristic algorithms and Genetic algorithm have significant role in optimization issues. This algorithm uses hybrid of Tabu Search and Genetic Algorithm to get more optimum results by using software cost estimation [12].

III. PROPOSED METHOD

In this subsection, we present the COCOMO, Bat Algorithm and our proposed method related to this research.

A. COCOMO

Boehm in 1981 introduced the research publication about the effort and schedule estimation model, called COCOMO [16]. Because of Boehm publication about COCOMO, this model becomes one of popular estimation models in 1980s.

Therefore, COCOMO in 1980's experienced difficulties to estimate the cost of software which is developed using new lifecycle processes and capabilities in new era. Therefore, there is a research to address problems on rapid development process model, non-sequential, reengineering, object oriented approaches and re-use driven approaches. Boehm introduced the new approach to COCOMO, called COCOMO II in 2000 [17]. This new approach is claimed better than COCOMO because there are some improvements in cost drivers and COCOMO II is also more accurate to solve the problems in software project estimation.

In COCOMO II, there are two models, namely Early Design and Post Architecture model. Early Design model is a high-level model in COCOMO II. This model has function to explore alternatives in architectural or do the development strategies. Meanwhile, the model which focuses on detail is called Post Architecture Model. This model works if the project is ready to be developed and maintain an existing system. The system should provide life-cycle architecture package. The detail information contains input of cost driver and cost estimates. In this research, we focus on using Post Architecture Model of COCOMO II. We implement (1) and (2).

$$PM = A \times (KLOC)^E \times \prod_{i=1}^{n} EM_i \qquad (1)$$

where

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

Where A and B are baseline calibration constants which was generated in calibrating in 161 projects which exist in COCOMO II dataset. Size refers to the size of the software project that is measured of Source Lines of Code in thousand (KLOC). Effort Multiplier is EM, meanwhile Scale Factor is SF.

B. Bat Algorithm

Bat algorithm is metaheuristic search that is researched and proposed by Xin-She Yang in 2010 [18]. This algorithm mimics bat behaviors using echolocation in searching prey, it can distinguish whether the object is prey, obstacle or another thing.

The need to update the bat position x_i and velocities v_i in a d-dimensional search space should be done in order to find the solution. The velocities v_i^t and new solution x_i^t at time step t are given by:

$$f_i = f_{min} + (f_{max} - f_{min})\beta$$

$$v_i^t = v_i^{t-1} + (x_i^t - x_*)f_i \qquad (3)$$

$$x_i^t = x_i^{t-1} + v_i^t$$

Where:

$\beta \in [0,1]$	: a random vector which obtained from uniform distribution
x_*	: the current global best location after comparing all of the n bats.

The Bat Algorithm also uses local search, it make the bat can convergence to the best solution. Equation 3 shows how to generated local walk randomly.

$$x_{new} = x_{old} + \epsilon A^t \qquad (4)$$

where:

$\epsilon \in [-1,1]$: random number

$A^t = A_i^t >$: the average loudness of all the bats at this time step.

Moreover, in every iterations process, the loudness A_i and the rate r_i of pulse emission must be updated. As the loudness usually decrease when a bat has found its prey, while the rate of pulse emission increases, we can choose the loudness as any value of convenience. For example we can use $A_{min} = 0$ and $A_0 = 1$, to indicate that a bat has just found the pray, we assuming $A_{min} = 0$ means that the bat has temporarily stop emitting any sound. The equation of the loudness A_i and the rate r_i are stated in (4).

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \qquad (5)$$

Where:

α and γ : constants

To move these bats towards optimal solution, their loudness and emission rates have to be updated so that the new solutions are improved.

C. Proposed Method

To optimize COCOMO II constants A and B, we use Bat algorithm to get optimal location of bats as constants A and B, by using update their location using (2), (3) and (4), then count the function by assigning it to (1) and finally count the minimum MMRE using (6).

The steps of proposed work are the following.

Input : Turkish dataset
Output : minimum MMRE of all data points

1. Start the Bat algorithm by randomizing virtual bats according to the amount of populations which have been set. The dimension of the space search is two dimension because we only search for two constants which act as bat location, namely constant A and constant B.

2. Count each estimated effort for each data points from the datasets by assigning each constants A and B which have been randomized at step 1, including Effort Multiplier and Scale Factor. Constant A and Effort Multiplier are assigned using (1) and constants B and Scale Factor are assigned using (2). The value of constants A and B are the same for 12 data points. In this step we get 12 estimated effort and we do this step for each individual bat.

3. Count the MRE for each data point by using (5). The actual effort of each data point in dataset is subtracted with estimated effort which is generated from step 2. In this step we get 12 MRE for each individual bat.

4. Count the individual evaluation MMRE using (6) by adding all 12 MRE which are produced from step 3 and divide it by 12. Run this step for all individual bats.

5. Compare the MMRE of all individual bat obtained by step 4 and find the minimum MMRE.

6. Iterate the algorithm and compare all MMRE for each iteration. The most minimum MMRE will be the optimal value of this running algorithm.

The steps of our proposed method are shown in Fig 1. The starting point is randomize virtual bats location as constants A and B, whereas the final point is obtaining the minimum MMRE.

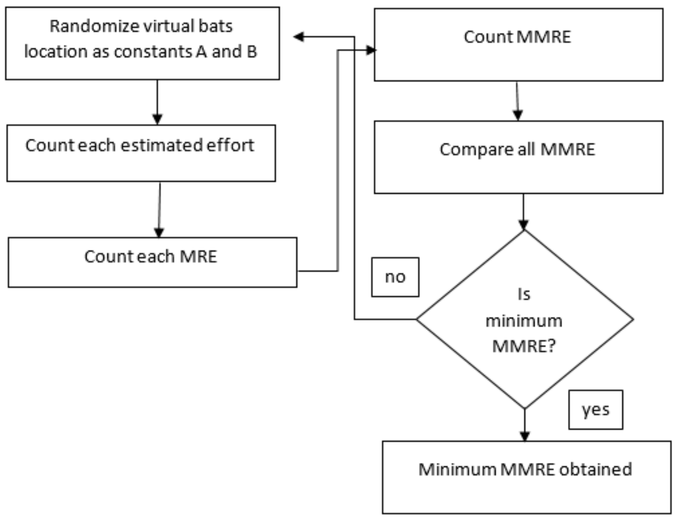

Fig. 1. The proposed method

IV. EXPERIMENT AND RESULT

A. *Evaluation Criteria and Dataset*

We use Magnitude Relative Error (MRE) and Mean Magnitude Relative Error (MMRE) to evaluate the result of proposed work.

$$MRE = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort} \times 100$$

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \qquad (6)$$

The proposed work is tested using Turkish projects dataset. Some of the datasets are shown in Table I. The Turkish project dataset contains 12 projects and some attributes, namely 5 Scale Factor, 17 Effort Multiplier, Line of Code and Actual Effort.

B. *Experimental Results*

The aim of this experiment is to optimize constants A and B of COCOMO II estimated effort using Bat algorithm. We hypothesize according to the previous works which have been done by using meta-heuristic algorithm where using this method, we can find the optimum constants A and B. Thus, with optimum constants we will get estimated effort which more closer to actual effort by evaluate it using its MMRE. In this experiment, we use Turkish project as dataset which contains 12 projects.

The parameters setting of Bat algorithm which are used from trial and error experiment, we record the best parameters to get the best result for this experiment. The parameter settings are population = 40, iteration = 2000, loudness = 0.5, pulse rate = 0.5, minimum frequency = 0 and maximum frequency = 2.

In the experiments, we get A = 4.5644 and B = -0.23931, compared with local calibration constants, A = 3.05, and B = 0.24. We also compare with COCOMO II constants, A = 2.94, and B = 0.91. The comparison can be seen in Table II.

The results in Table II shows that MMRE which is produced by Bat algorithm optimization have the smallest value compared with MMRE of optimization using local calibration and original COCOMO II, which is 41.64% better than local calibration, and 698.89% better than COCOMO II original constants.

We also present the MRE comparison of all methods in graphic which are shown in Fig 2. In that graphic, we can see that the MRE graph of COCOMO II original constants shows fluctuated MRE compared with the other two methods, which are Bat algorithm and local calibration. MRE of Bat algorithm shows the better stability graph among the other methods.

In Fig 2, we can also get the information that dataset which has been calibrated by COCOMO II and Turkish dataset have different characteristic. The effect of using the optimized constants in project which has different characteristic can make the estimated effort not reaching or closer to its actual effort, moreover it will make the estimated effort farther away like shown in the graph.

The experiment result which is shown in Table II and Fig. 2 indicate that estimated effort by assigning constants A and B which have been optimized using Bat algorithm is closer to actual effort of projects which exist in Turkish project dataset. We can use the new constants A and B to a new project which have similar characteristic with projects in the dataset.

Optimization using Bat algorithm can get optimum result, because Bat algorithm use both local search and global search in its searching, the local search can benefit the algorithm to convergence to the optimum point, and global search can benefit the algorithm to avoid local optima.

The optimal result of Optimizing COCOMO II constants A and B using Bat Algorithm is depending on the parameters that we set. It does not always give better results, sometimes it gives bad results, because the characteristic of meta-heuristic algorithm that will find the optimal solution by searching and finding.

In the experiment of proposed work, we set the parameters by trials and errors. If we get bad result, we abandon it, otherwise we keep it and compare it with the others which have good results. The results of our proposed of each data points does not always give the best results. We see the comparison of all the methods at data points 3, 6, 7, 9, and 12. Our proposed experiments show that the MRE results are not the best result among the other methods. This happens because the model that we proposed did not search for each optimal MRE, but we search only for optimal MMRE, which is why the MRE of each data point of our proposed work does not always give the best result among the other methods.

TABLE I. SOME OF DATA POINTS REPRESENTATION IN TURKISH SOFTWARE PROJECT DATASET

ID	1	2	3	4	5
PREC	Very High	Very High	Extra High	Extra High	Nominal
FLEX	Nominal	Nominal	Nominal	High	Nominal
RESL	Low	Low	Low	High	Low
TEAM	High	Very High	High	High	High
PMAT	Very Low	Very Low	Very Low	Nominal	Nominal
RELY	Nominal	High	High	High	Low
DATA	Nominal	Nominal	Very High	High	Nominal
CPLX	Low	Low	Low	Very High	Very High
RUSE	Low	Nominal	Nominal	Low	Nominal
DOCU	Low	Low	Low	Nominal	High
TIME	Very High	Very High	Very High	Nominal	High
STOR	High	High	Very High	Nominal	Nominal
PVOL	Nominal	Nominal	Nominal	High	Nominal
LOC	3000	2000	4250	10000	15000
AE	1.2	2	4.5	3	4

TABLE II. COMPARISON MRE AND MMRE OF BAT ALGORITHM, LOCAL CALIBRATION AND COCOMO IITABLE STYLES

Data	Bat Algorithm	Local Calibration	COCOMO II
1	27.70%	44.44%	190.67%
2	0.00%	6.87%	42.84%
3	39.13%	18.64%	106.76%
4	24.71%	151.20%	1032.58%
5	9.09%	166.88%	1478.89%
6	97.22%	89.05%	26.05%
7	64.39%	53.49%	14.43%
8	14.44%	200.28%	2841.79%
9	88.93%	28.31%	1553.36%
10	32.72%	102.47%	1499.90%
11	0.00%	22.34%	7.61%
12	12.70%	26.70%	2.75%
MMRE	**34.25%**	**75.89%**	**733.14%**

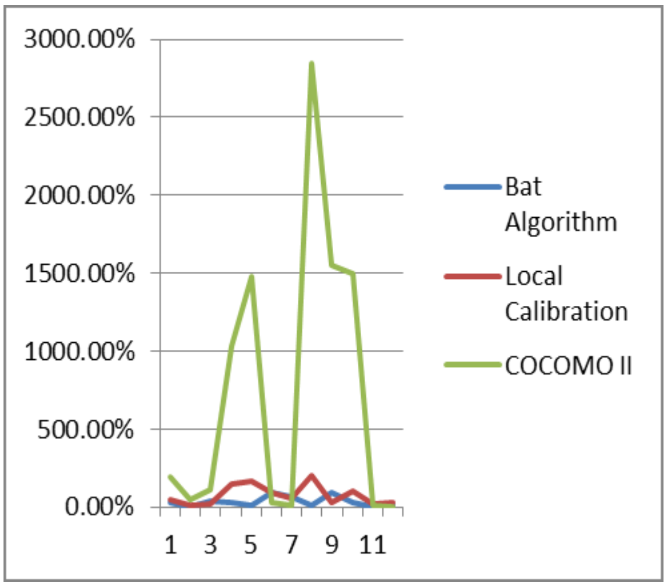

Fig. 2. Graphic MRE comparison of Bat Algorithm, Local calibration, and COCOMO II.

V. CONCLUSION

The proposed work, implementation of Bat Algorithm for COCOMO II optimization, generate better result compared with local calibration and original COCOMO II constants. We compared all models using Turkish project dataset. The experiments show that optimization using Bat algorithm can produce the smallest MMRE among the other methods. Optimization using Bat algorithm generates 41.64 % better than local calibration and 698.89 % better than COCOMO II original constants. Thus, we can assign constant A and constant B which are produced from optimization using Bat algorithm to the equation used in effort estimation. The effort estimation will produce the value which is more closer to actual effort of the projects in the dataset.

ACKNOWLEDGMENT

We would like to thank Information Management Lab, Department of Informatics, Faculty of Information and Communication Technology, Institut Teknologi Sepuluh Nopember for supporting the research.

REFERENCES

[1] M. A. Subandri and R. Sarno, "Cyclomatic Complexity for Determining Product Complexity Level in COCOMO II". *Information Systems International Conference (ISICO),* 2017. Procedia Computer Science, Volume 124, 2017, Pages 478-486.

[2] R. R. Putri, R. Sarno, D. O. Siahaan and A. Solichah. "Accuracy Improvement of The Estimations Effort in COCOMO II Based On

Logic Model of Fuzzy". *International Conference On Energy, Environment And Information System (ICENIS)*, 2016.

[3] M. Baiquni, R. Sarno, Sarwosri and Sholiq. "Improving the Accuracy of COCOMO II Using Fuzzy Logic and Local Calibration Method". *International Conference on Science in Information Technology (ICSITech),* 2017. https://doi.org/10.1109/ICSITech.2017.8257126.

[4] S. Aljahdali and A. F. Sheta, "Software Effort Estimation by Tuning COOCMO Model Parameters Using Differential Evolution,"2000.

[5] A. Malik, "An Analysis of Fuzzy Approaches for COCOMO II," no. April, pp. 68–75, 2013.

[6] K. Langsari and R. Sarno. "Optimizing COCOMO II Parameters using Particle Swarm Method". *International Conference on Science in Information Technology (ICSITech)*, 2017. https://doi.org/10.1109/ICSITech.2017.8257081.

[7] K. Langsari and R.Sarno. "Optimizing Effort and Time Parameters of COCOMO II Estimation using Fuzzy Multi-Objective PSO". *International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 2017. https://doi.org/10.1109/EECSI.2017.8239157.

[8] K. D. Febriyanti, R. Sarno and Y. A. Effendi, "Fraud Detection on Event Logs Using Fuzzy Association Rule Learning", *International Conference on Information & Communication Technology and Systems (ICTS),* 2017. https://doi.org/10.1109/ICTS.2017.8265661.

[9] S. Sadouni, "A New Model Of Software Cost Estimation Using Harmony Search," vol. 5, no. 3, pp. 47–55,2015.

[10] A. G. Saif, S. Abbas, and Z. Fayed, "Intelligent Water Drops (IWD) Algorithm for COCOMO II and COQUAMO Optimization," vol.I, 2015.

[11] R. K. Sachan *et al.*, "Optimizing Basic COCOMO Model using Simplified Genetic Algorithm," *Procedia - Procedia Comput. Sci.*, vol. 89, pp. 492–498, 2016.

[12] F. Soleimanian, "A New Approach by Using Tabu Search and Genetic Algorithms in Software Cost Estimation,"2015.

[13] S. Chalotra, S. K. Sehra, Y. S. Brar, and N. Kaur, "Tuning of COCOMO Model Parameters by using Bee Colony Optimization," vol. 8, no. July, 2015.

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

[15] Y. A. Effendi and R. Sarno, "Discovering optimized process model using rule discovery hybrid particle swarm optimization," *3rd International Conference on Science in Information Technology (ICSITech), pp. 97-103, 2017.* DOI: 10.1109/ICSITech.2017.8257092

[16] A. Puspaningrum and R. Sarno. "Optimization of COCOMO II Coefficients using Cuckoo Optimization Algorithm to Improve The Accuracy of Effort Estimation". *International Conference on Information & Communication Technology and Systems (ICTS),* 2017. https://doi.org/10.1109/ICTS.2017.8265653.

[17] M. Alajlan, "Optimization of COCOMO II Model for Effort and Development Time Estimation using Genetic Algorithms," no. March, 2016.

[18] X. Yang, "A New Metaheuristic Bat-Inspired Algorithm," pp. 1–10, 2010.