Journal of Engineering Science and Technology
Vol. XX, No. Y (Year) PPP - QQQ
© School of Engineering, Taylor's University

APPLYING TEACHING-LEARNING TO ARTIFICIAL BEE COLONY FOR PARAMETER OPTIMIZATION OF SOFTWARE EFFORT ESTIMATION MODEL

THANH TUNG KHUAT, MY HANH LE*

DATIC Laboratory,
The University of Danang, University of Science and Technology, Danang, Vietnam
*Corresponding Author: ltmhanh@dut.udn.vn

Abstract

Artificial Bee Colony inspired by the foraging behaviour of honey bees is a novel meta-heuristic optimization algorithm in the community of swarm intelligence algorithms. Nevertheless, it is still insufficient in the speed of convergence and the quality of solutions. This paper proposes an approach in order to tackle these downsides by combining the positive aspects of Teaching-Learning based optimization and Artificial Bee Colony. The performance of the proposed method is assessed on the software effort estimation problem, which is the complex and important issue in the project management. Software developers often carry out the software estimation in the early stages of the software development life cycle to derive the required cost and schedule for a project. There are a large number of methods for effort estimation in which COCOMO II is one of the most widely used models. However, this model has some restricts because its parameters have not been optimized yet. In this work, therefore, we will present the approach to overcome this limitation of COCOMO II model. The experiments have been conducted on NASA software project dataset and the obtained results indicated that the improvement of parameters provided better estimation capabilities compared to the original COCOMO II model.

Keywords: Software effort estimation, COCOMO II model, Artificial Bee Colony Algorithm, Teaching-Learning based Optimization.

1. Introduction

Metaheuristic optimization algorithms have become a ubiquitous choice for dealing with complicated problems which are difficult to solve by conventional

Nomenclatures

A	Multiplicative Constant (2.94 in COCOMO II)
actEffort	The actual effort
actTime	The actual time
B	Constant value (0.91 in COCOMO II)
C	Constant value (3.67 in COCOMO II)
D	Constant value (0.28 in COCOMO II)
d	The number of dimensions
E	An aggregation of five scale factors
e	Constant value (2.71828)
f	Fitness function
EM_i	The *ith* effort multiplier
estEffort	The estimate effort
estTime	The estimate time
I	The maximum number of iteration
MMRE	Mean of Magnitude of Relative Error
MRE	Magnitude of Relative Error
MRE_i	The *ith* Magnitude of Relative Error
p_i	The probability of each food source in ABC algorithm
P_s	The selection probability
PRED	Prediction level
SF_j	The *jth* scale factor
Size	The estimated size of software
SN	The size of population
T	Number of projects
TDEV	Development time
x_i	One solution of the problem

Greek Symbols

$\varphi_{i,j}$	A random number in the domain [-1, 1]

Abbreviations

ABC	Artificial Bee Colony
COCOMO II	Constructive Cost Model II
EAs	Evolutionary Algorithms
EM	Effort Multipliers
PA	Post-Architecture
PM	Person-Months
SF	Scale Factor
TLABC	Teaching-Learning Artificial Bee Colony
TLBO	Teaching-Learning-based Optimization

methods. These algorithms can be classified into various categories depending on the criteria being considered, such as population-based, iterative-based, stochastic, deterministic, etc. [1]. The population-based classification works with a set of solutions and trying to improve them. This group includes two typical classes of the algorithm: evolutionary algorithms (EAs) and swarm intelligence based algorithms. While EAs use mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection, swarm intelligence

based algorithms refer to the collective behaviour by means of local interactions among simple agents and with their environment. Based on the intelligent foraging behaviour of a honey bee swarm, Karaboga [2] proposed Artificial Bee Colony (ABC) algorithm for numerical optimization problems. The ABC has been used in many applications in several different fields such as training neural networks, solving the optimization problems encountered in electrical engineering, mechanical, civil engineering areas, and data mining [3]. The ABC algorithm shows promising performance in tackling optimization problems, but it still has some limitations in the speed of convergence and the quality of solutions. Teaching-Learning based optimization (TLBO) with fast speed has been widely employed to deal with kinds of optimization problems successfully [4 - 6]. Hence, an efficient hybrid algorithm with the advantages of TLBO and ABC, named TLABC, is proposed to improve the solution quality and accelerate the convergence speed. The effectiveness of the proposed algorithm is experimented on the software effort estimation problem.

Effort and cost estimation process in any software engineering project is an extremely important component. The success or failure of projects depends greatly on the accuracy of effort and schedule estimations. Both underestimated and overestimated effort can result in the serious issues [7]. Underestimation leads to a situation where a project's commitments cannot be accomplished because of a shortage of time and/or funds. In contrast, overestimation can cause the rejection of a project proposal. Many studies were figured out to find out the effective software cost estimation model. The introduction of the COCOMO II model has contributed significantly to the enhancement of accuracy in software cost estimation and currently this is one of the most commonly used models. However, this model relies on several constant values of parametric-based estimation equations. These constants have not been optimized, and thus the accuracy of estimations on projects is not high in comparison with the actual effort and time. In this paper, therefore, we propose the use of TLABC algorithm to optimize these parameters of COCOMO II models. The proposed approach increased the efficiency of COCOMO II model when experimenting on "NASA 93" projects [8].

The remainder of paper is structured as follows: Section 2 presents the COCOMO II model. The proposed approaches for parameters optimization of COCOMO II model are described in Section 3. Section 4 shows the empirical results and discussions. Conclusion and future work are represented in Section 5.

2. COCOMO II model

COnstructive COst MOdel II (COCOMO II) [9], which was developed in 1995 is a model that allows one to estimate the cost, effort, and schedule when planning a new software development activity. It takes qualitative inputs and produce quantitative results. In COCOMO II, the effort is represented as Person-Months (PM). A person month is the amount of time one person spends working on the software development project for one month [10].

COCOMO II has three sub-models including the Application Composition, The Early Design and Post-Architecture (PA) models [11]. This work considers the PA model, which is a detailed model that is used once the project is ready to

develop and sustain a fielded system. This model predicts software development effort by using the formula shown in Eq. (1).

$$PM = A \times Size^{E} \times \prod_{i=1}^{17} EM_i \quad (1)$$

where A is multiplicative constant having a value of 2.94. *Size*, which is the estimated size of software development, is the most important factor in calculating the effort of the software project and it is measured in Kilo Line of Code (KLOC). EM_i is one of a set of effort multipliers shown in Table 1. This is the seventeen Post-Architecture effort multipliers (EM) are used in the COCOMO II model to adjust the nominal effort. These multipliers are values of rating level of every multiplicative cost driver used to capture features of the software development that affect the effort to complete the project [10].

Table 1. Cost drivers for COCOMO-II PA model.

Driver	Symbol	Very Low	Low	Nominal	High	Very High	Extra High
RELY	EM_1	0.82	0.92	1.00	1.10	1.26	-
DATA	EM_2	-	0.90	1.00	1.14	1.28	-
CPLX	EM_3	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	EM_4	-	0.95	1.00	1.07	1.15	1.24
DOCU	EM_5	0.81	0.91	1.00	1.11	1.23	-
TIME	EM_6	-	-	1.00	1.11	1.29	1.63
STOR	EM_7	-	-	1.00	1.05	1.17	1.46
PVOL	EM_8	-	0.87	1.00	1.15	1.30	-
ACAP	EM_9	1.42	1.19	1.00	0.85	0.71	-
PCAP	EM_{10}	1.34	1.15	1.00	0.88	0.76	-
PCON	EM_{11}	1.29	1.12	1.00	0.90	0.81	-
APEX	EM_{12}	1.22	1.10	1.00	0.88	0.81	-
PLEX	EM_{13}	1.19	1.09	1.00	0.91	0.85	-
LTEX	EM_{14}	1.20	1.09	1.00	0.91	0.84	-
TOOL	EM_{15}	1.17	1.09	1.00	0.90	0.78	-
SITE	EM_{16}	1.22	1.09	1.00	0.93	0.86	0.80
SCED	EM_{17}	1.43	1.14	1.00	1.00	1.00	-

The exponent E in Eq. (1) is an aggregation of five scale factors (SF) that account for the relative economies or diseconomies of scale encountered for software projects of different sizes [9] and is computed as the following formula:

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

where, B is a constant having the value of 0.91. Each scale factor has a range of rating levels, from Very Low to Extra High. Each rating level has a weight which is presented in Table 2.

In addition to the effort, the software companies are also more interested in calculating the development time (TDEV) for projects. It is derived from the effort according to Eq. (3) and Eq. (4):

$$TDEV = C \times PM^{F} \quad (3)$$

$$F = D + 0.2 \times 0.01 \times \sum_{i=1}^{5} SF_i \quad (4)$$

The values of *C* and *D* for the COCOMO II schedule equation are obtained by calibration to the actual schedule values for the 161 project currently in the COCOMO II database and results are *C* = 3.67 and *D* = 0.28.

Table 2. Scale factor values for COCOMO II model.

Scale Factors	Symbol	Very Low	Low	Nominal	High	Very High	Extra High
PREC	SF_1	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	SF_2	5.07	4.05	3.04	2.03	1.01	0.00
RESL	SF_3	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	SF_4	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	SF_5	7.80	6.24	4.68	3.12	1.56	0.00

In software estimation, most practitioners use Mean of Magnitude of Relative Error (MMRE) and Prediction level PRED(x) to compute the error percentage. MMRE is the Mean of the Magnitude of Relative Error and it is a very common criterion used to appreciate software cost estimation models [12]. The Magnitude of Relative Error (MRE) for each observation *i* can be obtained as Eq. (5):

$$MRE_i = \frac{estimate_i - actual_i}{actual_i} \tag{5}$$

After that, the MMRE is the percentage of the absolute values of the relative errors, averaged over the *T* observations.

$$MMRE = \frac{100}{T} \times \sum_{i=1}^{T} MRE_i \tag{6}$$

On the other hand, PRED(*x*) is the percentage of projects for which the estimate falls within x% of the actual value. For instance, if PRED(30) = 80, this indicates that 80% of the projects fall within 30% error range. COCOMO's performance in practice is often measured in terms of PRED(30) [13]. Equation (7) shows how to calculate the value of PRED(*x*):

$$PRED(N) = \frac{100}{T} \times \sum_{i=1}^{T} \begin{cases} 1 & if\ MRE_i \leq \frac{N}{100} \\ 0 & otherwise \end{cases} \tag{7}$$

In this work, the values of *A*, *B*, *C*, and *D* will be tuned by using metaheuristics to attain the lowest value of MMRE and the highest value of PRED.

3. The algorithms for parameter optimization of COCOMO II model

3.1. The problem definition and fitness function

In the COCOMO II model, the values of *A*, *B*, *C*, and *D* are constant and they are not tuned following the actual effort and time of new software projects. Therefore, the accuracy of estimated activities for projects is not precise. In this paper, we propose the approaches to optimize these parameters of COCOMO II by using the historical software projects, ABC algorithm, and TLABC algorithm.

In the actual process of software development, developers often estimate the required effort and time for projects. Perfect estimation would give an MRE of 0% for both time and effort. In addition, the value of parameters which gives the estimated results on *T* projects approximating with the actual outcomes will be

desired. Hence, the lower value of MMRE on T projects is, the better results obtain. In this paper, the sum of time MMRE and effort MMRE is used in order to evaluate the fitness (f) of individuals in the population of solutions as Eq. (8).

$$f = \frac{100}{T} \times \sum_{i=1}^{T} \left(\frac{|estEffort_i - actEffort_i|}{actEffort_i} + \frac{|estTime_i - actTime_i|}{actTime_i} \right) \quad (8)$$

3.2. Artificial bee colony algorithm

Artificial Bee Colony algorithm proposed by Karaboga in 2005 for real parameter optimization is a recently introduced optimization algorithm and simulates the intelligent foraging behaviour of honey bee swarms [2]. The colony of artificial bees in the ABC consists of three categories of bees: *employed bees* associated with specific food sources, *onlooker bees* watching the dance of employed bees within the hive for making a decision to select a food source, and *scout bees* conducting random search to discover new sources. In the ABC algorithm, the number of food sources is equal to the number of employed bees and the number of employed bees equals to the number of onlooker bees as well. The food source of which the nectar is abandoned by the employed or onlooker bees is replaced with a new food source by the scouts. In other words, the employed bee or onlooker bee whose food source has been exhausted becomes a scout bee. Main steps of the ABC algorithm simulating behaviour of members in the colony are given in Algorithm 1.

Algorithm 1. Artificial bee colony algorithm

Initialize all parameters
Generate the initial population of solutions $x_{i,j}$, $i = 1...SN$, $j = 1...d$
Evaluate the fitness (f_i) for each individual of the population
while stopping criteria are not met **do**
- Produce new solution $v_{i,j}$ for the employed bees by using Eq. (10) and compute their fitness value
- Apply the greedy selection process
- Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ by Eq. (11)
- Generate the new solutions $v_{i,j}$ for the onlookers from the solutions $x_{i,j}$ chosen depending on $P_{i,j}$ and assess their fitness value
- Apply the greedy selection process
- Identify the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{i,j}$ by using Eq. (9)
- Memorize the best solution achieved so far

end while
return best solution

3.2.1. Initialization of swarm

The ABC algorithm has three major parameters: the number of food sources (the number of employed bees or the size of population), the quantity of test subsequent to which a food source is treated to be deserted and the termination criteria (maximum number of cycles or fitness function has reached to predefined value). At the first step, the ABC generates a randomly distributed initial population of SN solutions, where SN denotes the size of population. Each

solution x_i (i = 1, 2,..., SN) is a d-dimensional vector. In the parameter optimization of COCOMO II model, d is the number of optimization parameters (d = 4). Each solution is generated by using Eq. (9).

$$x_{i,j} = x_j^{min} + rand[0,1] \times (x_j^{max} - x_j^{min}) \quad (9)$$

Here *rand*[0,1] is a function that gives an equally distributed random number in the range of [0,1].

3.2.2. Employed bees phase

In the employed bees phase, artificial employed bees seek new food sources having more nectar within the neighbourhood of the food source in their memory. The ABC uses Eq. (10) in order to generate a candidate food position from the old one (x_i).

$$v_{i,j} = x_{i,j} + \varphi_{i,j} \times (x_{i,j} - x_{k,j}) \quad (10)$$

where, $k \in \{1, 2,...,SN\}$ ($k \neq i$) and $j \in \{1, 2,...,d\}$ are randomly chosen indexes. $\varphi_{i,j}$ is a random number in the domain [-1, 1].

After producing the new food source, its fitness is calculated and a greedy selection is applied between it and its parent. If new food source has higher fitness value, it will replace the existing one.

3.2.3. Onlooker bees phase

In the onlooker bees phase, artificial onlooker bees carry out the probabilistic selection of their food sources depending on the information provided by the employed bees. For this purpose, a fitness based selection technique can be employed, such as the roulette wheel selection method [3]. The probability of each food source is decided by using its fitness (f_i) as Eq. (11).

$$p_i = \frac{f_i}{\sum_{k=1}^{SN} f_k} \quad (11)$$

After a food source for an onlooker bee is probabilistically chosen, a neighbourhood source is determined by using Eq. (10), and its fitness value is computed. As in the employed bees phase, a greedy selection is applied between two sources.

3.2.4. Scout bees phase

Employed bees whose solutions cannot be enhanced through a predetermined number of trials, called "limit", become scouts and their sources are considered to be exhausted. Then, the scouts begin to search for new solutions randomly by using Eq. (9) and replace for the old ones.

3.3. Teaching-learning based optimization

Teaching-Learning based optimization algorithm which proposed by Rao et al. [14] is one of the most recently developed meta-heuristics. This algorithm is the population-based algorithm inspired by learning process in a classroom. For the TLBO, the population is considered as a group of learners or a class of learners. The search process contains two phases: Teacher phase and Learner phase.

3.3.1. Teacher phase

In the teacher phase, learners get knowledge from a teacher. In the entire population, the best solution is considered as the teacher ($X_{teacher}$). In this phase, the teacher tries to improve the results of other individuals (X_i) by increasing the average result of the classroom (X_{mean}) towards his/her level [14]. The solution is updated according to the difference between the existing and the new mean given by Eq. (12).

$$X_{new} = X_i + r_i \times (X_{teacher} - T_f \times X_{mean}) \tag{12}$$

where, T_f is a teaching factor that decides the value of mean to be changed, and r_i is a random number in the range [0, 1]. The value of T_f can be either 1 or 2, which is again a heuristic step. In addition, X_{new} and X_i are the new and existing solutions of the i^{th} learner respectively.

3.3.2. Learner phase

In the learner phase, learners try to increase their knowledge by interacting with others. A learner interacts randomly with other learners with the help of group discussions, presentations, formal communications, etc. [14]. A learner learns something new if another learner has more knowledge than him or her. Learner modification is represented as follows:

$$X_{new} = X_i + r_i \times (X_j - X_k) \quad \text{if } f(X_j) < f(X_k) \tag{13}$$

$$X_{new} = X_i + r_i \times (X_j - X_k) \quad \text{if } f(X_k) < f(X_j) \tag{14}$$

where X_k and X_j ($j \neq k$) are two students chosen randomly in the population, and f is the fitness function. If the new solution X_{new} is better, it is accepted in the population. The algorithm will continue until the termination condition is met.

3.4. Applying teaching-learning to artificial bee colony

A limitation of the ABC algorithm is that the convergence rate of the algorithm is poor. This issue arises from the stochastic variation process in which new solutions are produced from the parent solutions. While producing a new solution (v_i), changing only one parameter of the parent solution (x_i) causes a slow convergence rate. In order to overcome this problem, the advantages in the convergence speed of TLBO are combined into the employed bees and the onlooker bees phases when generation new solutions. In particular, the teacher mechanism of the TLBO is applied for the employed bees phase and the learner phase will be combined with the onlooker bees phase when producing new

individuals. We suppose that P_s is the probability of employing ABC strategy, and the TLBO strategy is employed with the probability of 1 - P_s.

At the beginning, the ABC operator with slow convergence ability is employed to expand the solution space. The larger P_s value is employed the higher probability of employing ABC operator. In the later stages of the search process, the TLBO operator is adopted to increase the convergence speed and get the best solution. To strengthen the exploitation ability, P_s value is decreasing in the evolution process. This paper proposes the selection probability as Eq. (15).

$$P_s(i) = \sin(e^{\frac{-3\times i}{I}}) \tag{15}$$

where i and I denote the current and the maximum number of iteration respectively. Figure 1 illustrates an example of P_s assigned for the number of cycles of 5000.

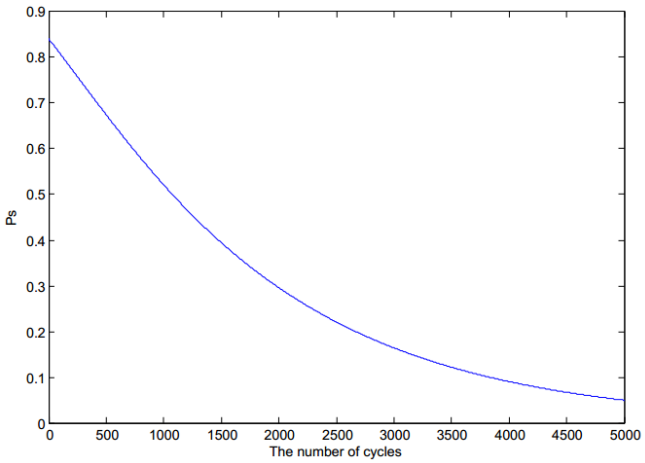

Fig. 1. Probability graph of P_s assigned for the number of cycles of 5000.

3.4.1. Employed bees phase

The teacher phase of the TLBO is combined with the ABC strategy when generating a new solution. The details of employed bees phase are presented as Algorithm 2.

Algorithm 2. Applying teaching for the employed bees phase

1. **if** *rand*(0, 1) < P_s
2. Select j in {1..d}, k in {1...SN} randomly
3. $v_{i,j} = x_{i,j} + \varphi_{i,j} \times (x_{i,j} - x_{k,j})$
4. **else**
5. **for** j = 1 to d **to**
6. $v_{i,j} = x_{i,j} + rand(0,1) \times (x_{teacher,j} - T_f \times x_{mean,j})$
7. **end for**
8. **end if**

3.4.2. Onlooker bees phase

The learner mechanism of the TLBO is applied for the onlooker bees phase when producing a new solution. The details of onlooker bees phase are described in Algorithm 3

Algorithm 3. Applying learning for the onlooker bees phase

if $rand(0, 1) < P_s$
    Select j in {1..d}, k in {1...SN} randomly
    $v_{i,j} = x_{i,j} + \varphi_{i,j} \times (x_{i,j} - x_{k,j})$
else
    Select two individuals x_t, and x_k ($t \neq k$) in the population randomly
    for $j = 1$ to d **to**
        if $f(x_t) < f(x_k)$
            $v_{i,j} = x_{i,j} + rand(0,1) \times (x_{t,j} - x_{k,j})$
        else
            $v_{i,j} = x_{i,j} + rand(0,1) \times (x_{k,j} - x_{t,j})$
        end if
    end for
end if

4. Experiments

The main objective of the experiment carried out is to reduce the uncertainty of current COCOMO II post architecture coefficients (*A*, *B*, *C*, and *D*) and to achieve the best software effort estimation results being equivalent to the actual effort. Furthermore, the convergence speed of both the ABC and TLABC will be considered.

Experiments have been conducted on "NASA 93" dataset [8], in which 65 projects were used as training data to optimize the parameters for COCOMO II model and the other 28 projects were used for testing the performance of this model after optimizing coefficients. In this experiment, the configuration parameters for both the ABC and TLABC are that the number of cycles is 5000, colony size is 200, and the value of "limit" is 200.

Experiments were carried out 25 times on the test sets and the best combinations of coefficients for each algorithm were reported. The optimized COCOMO II PA coefficients by using the ABC are A = 3.933, B = 0.864, C = 2.973, and D = 0.356 meanwhile the results using the TLABC are A = 4.062, B = 0.857, C = 2.938, and D = 0.357. The convergence of the model parameters after each generation is described in Fig. 2 for the ABC and Fig. 3 for the TLABC algorithm.

As seen from Fig. 2 and Fig. 3, the TLABC algorithm converged faster than the ABC. In general, the convergence speed of the ABC was improved when applying Teaching-Learning mechanism.

In order to evaluate the accuracy of predictions, the MMRE and PRED were employed. The accuracy of an estimation technique is proportional to the PRED and inversely proportional to the MMRE. In this experiment, the MMRE, PRED (10), PRED (20), PRED (30), and PRED (40) were used as evaluation criteria. To compute the values of MMRE and PRED, we use formulas as presented in

Section 2 with the optimized coefficients as mentioned above. These results are shown in Table 3.

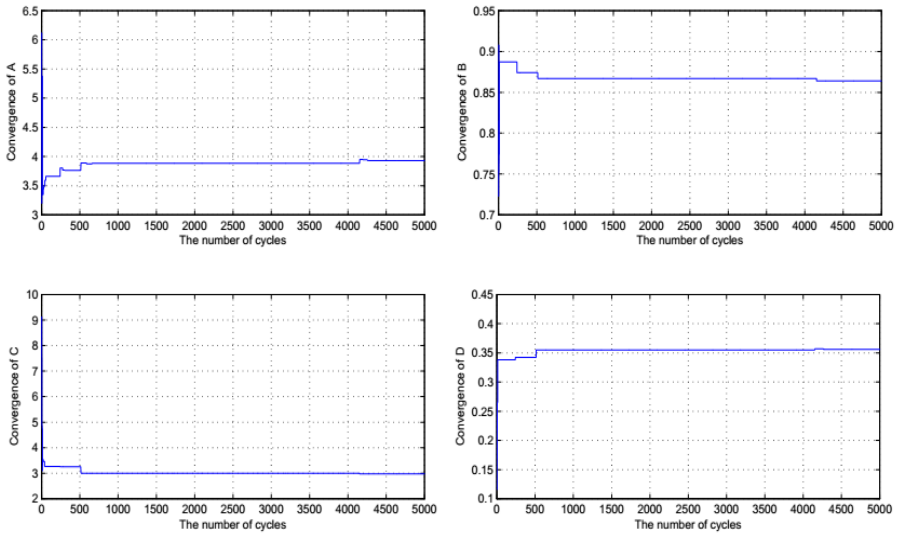

Fig. 2. Convergence of the model parameters *A, B, C* and *D* using the ABC.

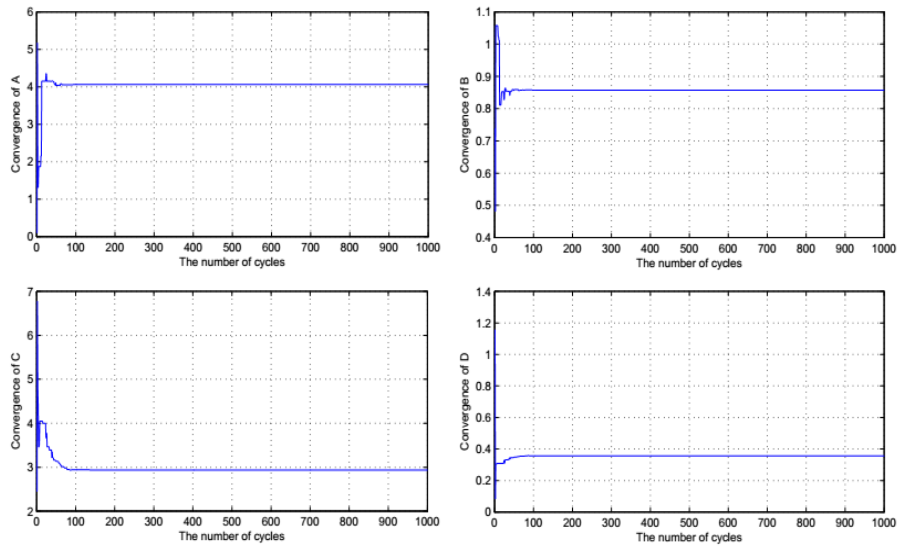

Fig. 3. Convergence of the model parameters *A, B, C* and *D* using the TLABC.

From Table 3, it can be seen that the estimated results of COCOMO II model after optimizing the parameters were much better than that of original COCOMO II model in terms of both effort and time. In particular, the MMRE of original COCOMO II model decreased more than 6% when its parameters were optimized by using ABC and TLABC algorithms. We can also see that the results using the TLABC are higher than those using the ABC. This may be clearly demonstrated

by the value of PRED(10) for time of the estimation model employing the TLABC is more outstanding when compared to the result of the model using the ABC algorithm.

Table 3. The results of evaluation criteria for approaches.

Criteria	Effort			Time		
	ABC (%)	TLABC (%)	COCOMOII (%)	ABC (%)	TLABC (%)	COCOMOII (%)
MMRE	21.68	21.57	27.50	8.66	8.29	14.71
PRED(10)	42.86	42.86	17.86	64.29	71.43	17.86
PRED(20)	62.29	64.29	50.00	96.43	96.43	50.00
PRED(30)	75.00	78.57	67.86	99.10	100	67.86
PRED(40)	82.14	82.14	82.14	100	100	82.14

5. Conclusion and future work

In this work, we figured out the performance of standard version and modified one by applying the Teaching-learning mechanism for the Artificial Bee Colony algorithm. We also compared the performances of modified model against the parameter optimization problem of COCOMO II model. From the obtained results, it can be seen that the TLABC has outperformed the ABC in terms of the convergence speed and the solution quality. It can be concluded that the COCOMO II model after optimizing parameters by using the TLABC obtained promising results for the software effort estimation.

Future work intends to apply the TLABC algorithm for Agile software effort estimation problem. We also employ the various nature-inspired algorithms to optimize the parameters of COCOMO II model.

References

1. Karaboga, D.; and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm. *Journal of Global Optimization*, 39(3), 459-471.
2. Karaboga, D. (2005). An idea based on Honey Bee Swarm for numerical optimization. *Technical Report*. Erciyes University, Engineering Faculty, Computer Engineering Department.
3. Karaboga, D.; Gorkemli, B.; Ozturk, C.; and Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, 42(1), 21-57.
4. Rao, R.V.; Patel, V.; and Vakharia, D.P. (2012). Teaching-learning-based optimization: An optimization method for continuous non-linear large scale problems. *Information Sciences*, 183, 1-15.
5. Rao, R.V.; and Patel, V. (2013). Multi-objective optimization of two stage thermoelectric cooler using a modified teaching-learning-based optimization algorithm. *Engineering Applications of Artificial Intelligence*, 26, 430-445.

6. Rao, R.V.; and Kalyankar, V.D. (2013). Parameter optimization of modern machining process using teaching-learning-based optimization algorithm. *Engineering Applications of Artificial Intelligence*, 26, 524-531.
7. Jones, C. (2003). Why flawed software projects are not cancelled in time. *Cutter IT Journal*, 10(12), 12-17.
8. Menzies, T. (2015). The Tera-PROMISE Repository for COCOMO 93. Retrieved August 8, 2016, from http://openscience.us/repo/effort/cocomo/nasa93.html
9. Boehm, B.; Clark, B.; Horowitz, E.; Westland, C.; Madachy, R.; and Selby, R. (1995). Cost Models for Future Software Life Cycle Processes: COCOMO 2.0. *Annals of Software Engineering*, 1(1), 57-94.
10. Abts, C.; Clark, B.; Devnani-Chulani, S.; Horowitz, E.; Madachy, R.; Reifer, D.; Selby, R.; and Steece, B. (1998). *COCOMO II model definition manual*. Center for Software Engineering, University of Southern California.
11. Clark, B.; Devnani-Chulani, S.; and Boehm, B. (1998). Calibrating the COCOMO II post-architecture model. *Proceedings of 20th International Conference on Software Engineering*. Kyoto, Japan, 477-480.
12. Nassif, A.B.; Capretz, L.F.; and Ho, D. (2010). Software estimation in the early stages of the software life cycle. *Proceedings of International Conference on Emerging Trends in Computer Science, Communication and Information Technology*. Maharashtra, India, 5-13.
13. Chen, Z.; Menzies, T.; Port, D.; and Boehm, B. (2005). Feature subset selection can improve Software Cost Estimation Accuracy. *Proceedings of the 2005 workshop on Predictor Models in Software Engineering*. New York, USA, 1-6.
14. Rao, R.V.; Savsani, V.J.; and Vakharia, D.P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43, 303-315.