TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 3, June 2021, pp. 817~828
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i3.16907

Parameter tuning of software effort estimation models using antlion optimization

Marrwa Abd-AlKareem Alabajee, Najla Akram AlSaati, Taghreed Riyadh Alreffaee
Department of Software, College of Computer Science and Mathematics, University of Mosul, Iraq

Article Info

Article history:

Received Jun 6, 2020
Revised Oct 5, 2020
Accepted Oct 15, 2020

Keywords:

Antlion optimization algorithm
Parameter tuning
Software effort estimation
The COCOMO model

ABSTRACT

In this work, the antlion optimization (ALO) is employed due to its efficiency and wide applicability to estimate the parameters of four modified models of the basic constructive cost model (COCOMO) model. Three tests are carried out to show the effectiveness of ALO: first, it is used with Bailey and Basili dataset for the basic COCOMO Model and Sheta's Model 1 and 2, and is compared with the firefly algorithm (FA), genetic algorithms (GA), and particle swarm optimization (PSO). Second, parameters of Sheta's Model 1 and 2, Uysal's Model 1 and 2 are optimized using Bailey and Basili dataset; results are compared with directed artificial bee colony algorithm (DABCA), GA, and simulated annealing (SA). Third, ALO is used with Basic COCOMO model and four large datasets, results are compared with hybrid bat inspired gravitational search algorithm (hBATGSA), improved BAT (IBAT), and BAT algorithms. Results of Test1 and Test2 show that ALO outperformed others, as for Test3, ALO is better than BAT and IBAT using MAE and the number of best estimations. ALO proofed achieving better results than hBATGSA for datasets 2 and 4 out of the four datasets explored in terms of MAE and the number of best estimates.

Corresponding Author:

Marrwa Abd-AlKareem Alabajee
Department of Software
Mosul University
Al Majmoaa Street, Mosul, Iraq
Email: marrwa_zedan@uomosul.edu.iq

1. INTRODUCTION

With over the last few decades, software systems have spread outstandingly and this spread is very likely to go on much swifter in the future. This is due to the rapid growth in scientific, industrial, and commercial technologies all around the world [1]. Starting a project with an acceptably estimated size or effort enables the manger to gain confidence regarding future activities, this is due to the fact that most of the decisions made throughout the development are affected by the initial estimations. So, the process of cost or effort estimations is considered as one of the most decisive stages of planning and managing software projects [2]. The importance of estimating software effort during early stages of software development has been pointed out so many times. But then again, estimating the software effort is considered to be very hard at the beginning of a project, this is because people productivity differs, so very little is known about the software being developed [1].

As the software development cost or effort is progressively rising, software has grown to be the major cost taken by a system. Many projects such as National Aeronautics and Space Administration (NASA) or Air Force estimates software development cost to reach 50% of the total cost required. This is due the high complexity encountered in the NASA software projects' systems, either hardware or software [3]. All this led

the way towards the development of a model capable of predicting software effort, and although there are various accurate models for computing project's effort, there is still a strong challenge for more accurate and precise models. Of these powerful models is the constructive cost model (COCOMO) that estimates the effort of any project during the early stage of the project. This can lead to minimize the overall cost of the project [1].

The process of effort estimation is affected by a large number of parameters; these parameters are estimated using different techniques. The aim of this work is to use the antlion optimization algorithm, attributable to its efficiency, in tuning the parameters of the COCOMO model variants, the first is the basic COCOMO model and the other four are its extensions. After parameter tuning, the obtained results are compared with the other methods. Most of the work related to this field goes back to 2010, when Aljahdali and Sheta [3] suggested the use of differential evolution (DE) to estimate the COCOMO model parameters, these models were tested using NASA software project dataset.

In 2012, Singh and Misra [4] explored the crisp input effect with genetic algorithms (GAs), and applied a modified version of the COCOMO model to NASA dataset. In the same year, Kundu and Sethi [5] recommended the use of simulated annealing (SA) to optimize the coefficients of COCOMO II model aiming for more accurate effort estimations. Later in 2013, Dhiman and Diwaker [6] also used GAs to optimize the coefficients of COCOMO II model in order to acquire accurate estimations.

Ghatasheh *et al*. [7], explored the use of the Firefly algorithm in 2015 to optimize the parameters of three COCOMO based models. Also, in 2015, Gupta and Sharma [8] submitted a new calibrated intermediate COCOMO model developed using the bat algorithm. The model generated new optimized coefficients, and results showed that the optimized coefficients gave better results for all project types in terms of mean magnitude of relative error (MMRE) when compared to coefficients gained using regression analysis.

Khuat and Le [9] in 2016 used artificial bee colony algorithm for parameter tuning according to the actual effort. Their work was verified with NASA software dataset and was compared to some existing models. In the same year, Bardsiri and Dorosti [10] increased the accuracy of COCOMO estimation by a hybrid model that combined bee colony algorithm with the COCOMO estimation method. Their method gave more efficient coefficient comparative to the basic COCOMO, finding better coefficients can greatly maximizes the method's efficiency. Also, in 2016 Girotra and Sharma [11] considered cost driver and the issue of inaccurate and ambiguous selection of values which leads to inaccurate effort estimations, and they showed that a small change in the selection of COCOMO cost drivers can cause significant improvements in metrics such as MMRE. Whereas in 2017, A-Srhan, *et al.* [1] established a hybrid cuckoo search algorithm and genetic algorithm called (CSGA) for parameter estimation. A NASA software project dataset was used in the experiments. Results show that CSGA enhanced the effort estimation accuracy.

In 2018 Nandal and Sangwan [12] introduced a hybrid bat inspired gravitational search algorithm method called (BATGSA) to optimize the COCOMO model. In the same year, Khatibi and Bardsiri [13] suggested a combined model for effort estimation. The model was based on particle swarm optimization algorithm with a linear regression method to optimally discover coefficient. Later in 2018 Dizaj and Gharehchopogh [14] improved GAs with bat algorithm to study the influence of qualitative factors and false variables on the total cost estimation. Their model was explored and tested using four datasets with seven criteria; results showed that the model improved the accuracy of cost estimation.

To give a precise estimated cost for project development, in 2019 Venkataiah *et al*. [15] suggested the implementation of hybrid methodology for tuning parameters of COCOMO model. To check the efficiency of the presented model, they used IBMDPS, COCOMO NASA 2 and DESHARNAIS and COCOMO 81. As for the employed optimization method, ALO algorithm has been successfully applied in many areas such as tuning the parameter of control devices in systems of generators' excitation for the model of TAFM in 2018 by Špoljarić and Pavić [16]. In this work, the ALO algorithm is employed along with heavy comparisons in opposite to related work aiming to cover possible gaps; using five models instead of only one or three, the same also goes for datasets engaged in testing and comparisons, as five different sized datasets were used to carry out inclusive comparisons among ALO and the other methods.

2. RESEARCH METHOD

This section includes problem description of software effort estimation models, there are many models suggested for estimating software effort, all of them were derived from the COCOMO Model, in this section, the software effort estimation models that were utilized in this paper will be described as well as the mathematical equation of them. In addition, this section includes an explanation of the antlion optimization algorithm and describes the methodology of this algorithm and explains how can the traps of the antlions affect the random walk of ant and how can antion building the traps to make the trapped ant sliding down towards the center of the pits, this mechanism modeled by mathematical equations described below. Finally, the last subsection represents the mechanism of elitism of this algorithm.

2.1. Software effort estimation models

Software cost estimation is the practice of predicting the required effort for developing a project. Such estimation gives the impression of simplicity, but in reality, it is very difficult and complex. Costs for software projects depend largely on the project's nature and characteristics, while the estimation accuracy depends merely upon the amount of reliable information gained regarding the developed product [5].

Scientific efforts are being carried out for developing new techniques to estimate software cost. Nearly all estimation models for software cost are algorithmic and expert judgment based. Accuracy modelling affects estimation accuracy, which is why finding good models for software estimation is the greatest significant objective for software engineers. In the core of these models is the COCOMO (constructive cost model), this model is the most frequently used due its simplicity in estimating the person-month effort for projects at various development stages [5].

The COCOMO model in (1) was first developed in 1984 by Boehm [17]. It has been considered to be empirical due to the enormous amount of data used in its development; these data are taken from several projects. In addition, it is found that many project managers employ the COCOMO model; this is because its details are available unlike other models [18, 19].

$$E = a(SIZE)^b \quad (1)$$

Parameter values (*a*) and (*b*), rely principally on the software project class. Software projects were classified based on the complexity of the project into three categories: organic, semidetached, and embedded. The model helps is defining mathematical equations that identify the cost, schedule and quality of a software product [18].

This work represents an attempt to optimize the parameters of five variations of the COCOMO model. The first is the basic COCOMO model given in (1). The other two are modifications of the basic COCOMO model proposed by Sheta [20], both modified models consider the methodologies (ME) to linearly affect effort. One of them is Sheta's Model 1 given in (2) and is named (Model I) in this work, the other is Sheta's Model 2 given in (3), and named (Model II) here [7].

$$E = a(Size)^b + c\ (ME) \quad (2)$$

$$E = a(Size)^b + c(ME) + d \quad (3)$$

The last two models are proposed by Uysal [21], one of them contain five parameters a, b, c, d and e called Uysal's Model 1, and is called (Model III) here, as in (4) [21].

$$E = a(Size)^b + c.ME^d + e \quad (4)$$

The other model called Uysal's Model 2 and called (Model IV) here, is presented as in (5)

$$E = a(Size)^b + c.ME^d + e.\ln(ME) + f.\ln(Size) + g \quad (5)$$

In this work, an attempt is conducted to optimize parameters (*a, b, c, d, e, f, and g*) using the antlion optimization (ALO) algorithm.

2.2. The antlion optimization algorithm

Insects like antlions belong to a group in the family of myrmelentidae. The two main stages of its lifecycle are larval and adult stages. The antlion larva leaves trails in the sand in the search of a good location to construct its trap, which is why it is called "doodlebug". Antlions make a pit in the sand to hide inside it during hunting as shown in Figure 1 (a). Figure 1 (b) depicts the slipping of the prey towards the bottom, the antlion instantly grabs it. If it tries to escape, the antlion tosses some sand to the edge of the pit so that the pray slides into the lowermost of that pit. The larva also weakens the pit's sides, forcing them to drop and take the prey with them [22]. The ALO algorithm inspiration comes from the foraging behaviour of the antlion's larvae [23].

2.2.1. ALO methodology

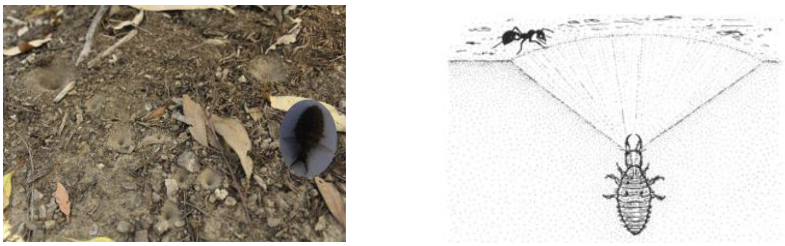
Modelling the relationship between antlions and ants requires ants to go through the search space, where antlions are permitted to hunt them and thus develop their fitness to traps. In nature, ants move in a stochastic manner in search for food, thus a stochastic function (r(t)) can be defined as in (6), and a random walk is suitable for modelling the movement of ants as given in (7) [24].

$$r(t) = \begin{cases} 1 \text{ if rand} > 0.5 \\ 0 \text{ if rand} \leq 0.5 \end{cases} \quad (6)$$

where *rand* is a random number made with uniform distribution in the interval of [0, 1].

$$X(t) = [(0, \text{cumsum}(2r(t_1) - 1, \ldots, \text{cumsum}(2r(t_n) - 1)] \quad (7)$$

where *Cumsum* is calculates the cumulative sum, *n* is the maximum number of iteration, and *t* shows the iteration,



(a) (b)

Figure 1. Behaviour of hunting [23]

A matrix M_{Ant} is used to store the position for all ants as in (8); this matrix will be utilized throughout optimization [23].

$$M_{Ant} = \begin{bmatrix} A_{1,1} & A_{1,2} \cdots & A_{1,d} \\ \vdots & \vdots \cdots & \vdots \\ A_{n,1} & A_{n,2} \cdots & A_{n,d} \end{bmatrix} \quad (8)$$

Ants are noted to be similar to the particles in PSO or individuals in GA, an ant's position denote a particular solution parameter. The M_{Ant} Matrix is used to register the post of all ants throughout optimization. To evaluate each ant, a function of fitness should be employed, matrix M_{OA} gathers the fitness values for all ants as in (9) [23]. Moreover, antlions are also assumed to be hiding someplace in the search space, and with the aim of saving their positions and fitness values, two matrices are used, $M_{Antlion}$ and M_{OAL} [23, 25].

$$M_{OA} = \begin{bmatrix} f(A_{1,1}A_{1,2} & \ldots & A_{1,d}) \\ \vdots & \ldots & \cdots \\ f(A_{n,1}A_{n,2} & \ldots & A_{n,d}) \end{bmatrix} \quad (9)$$

Basically, all random walks are established using (7), the update of positions for ants is done using the random walk at each and every stage of optimization. Nevertheless, updating position of ants cannot be directly accomplished using (7). So, to restrict the random walks of ants in the search space, (10) is used to normalize them and it must be applied in all iterations to ensure that the random walk occur inside the search space [22, 23]. Where, a_i is the minimum of random walk for the i^{th} variable, b_i is the maximum of random walk in i^{th} variable, C_i^t is the minimum of i^{th} variable at i^{th} iteration, d_i^t is the maximum of i^{th} variable at i^{th} iteration.

$$X_i^t = \frac{(X_i^t - a_i) \times (d_i - C_i^t)}{(d_i^t - a_i)} + C_i \quad (10)$$

2.2.2. Pits and traps of antlions

Antlions' traps affect random walks of ants; this assumption is modelled mathematically using two proposed equations in (11) and (12). These two equations show that ants walk randomly in a hyper sphere expressed by C and D vectors nearby a selected antlion [22, 25]. Where $Antlion_j^t$ is the position of selected j^{th}Antlion at t^{th} iteration. The selection of antlions is based on their fitness using the roulette wheel. This mechanism gives high chances to the fitter antlions to catch ants. Figure 2 illustrates how ants are expected to be trapped in only one particular antlion [23].

$$C_i^t = \text{Antlion}_j^t + C^t \quad (11)$$

$$D_i^t = Antlion_j^t + D^t \quad (12)$$

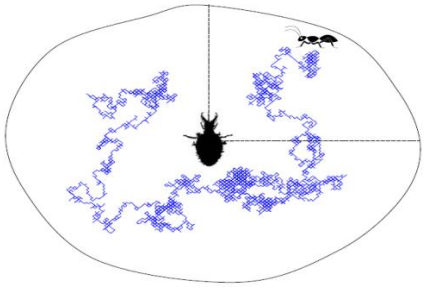


Figure 2. Random walk of an ant inside the trap

2.2.3. Sliding towards antlions

Antlions are now capable of building traps relatively to their fitness, where ants are required to move in random. Yet, once antlions recognize that an ant is trapped, they begin to throw sands away from the centre of the pit. This act slides down the escaping trapped ant, and to model this behaviour, the radius of the hyper-sphere for ants' random walks is reduced according to the ratio calculate by [22] and given in (13). Based on this ratio, two reductions are proposed as in (14) and (15) [26, 27].

$$I = 10^w . \frac{t}{T} \quad (13)$$

where *t* resembles current iteration, *T* gives the maximum number of iterations, and *w* is a constant defined based on the current iteration (w=2 when t>0.1T, w=3 when t>0.5T, w=4 when t>0.75 T, w=5 when t>0.9T, and w=6 when t>0.95T). Basically, the constant *w* can adjust the accuracy level of exploitation [23].

$$C^t = \frac{c^t}{I} \quad (14)$$

$$d^t = \frac{d^t}{I} \quad (15)$$

Hunting comes to an end when an ant arrives at the bottommost area of the pit and is imprisoned by the jaws of the antlion. Subsequently, an antlion starts to drag the ant into the sand and ingest its body. To simulate such a process, an assumption has to be made that prey catching take place when ants grow fitter (sink in sand) than the correspondent antlion. Accordingly, the antlion has to enhance its chance of catching new prey by updating its position to the latest known position of the hunted ant; this is given in (16) [23]. Where *t* is the current iteration, Ant_i^t *:* is the position of i^{th} ant at t^{th} iteration.

$$Antlion_j^t = Ant_i^t \text{ if } f(Ant_i^t) > f(Antlion_j^t) \quad (16)$$

2.2.4. Elitism

Best gained solutions are at risk of being lost through subsequent iterations; that is why elitism is needed. It allows best solutions to be kept through the iterations of the process. Being the fittest antlion, elite can affect the movements of each ant during all iterations. Hence, every ant is presumed to walk randomly around the antlion using roulette wheel and elitism concurrently as in (17) [23, 27].

$$Ant_i^t = \frac{r_a^t + r_e^t}{2} \quad (17)$$

2.2.5. ALO algorithm for parameter tuning

The workflows of the ALO algorithm can be graphically represented to describe the sequential or concurrent flows of activities of the algorithm in parameter tuning as in Figure 3. This figure shows the Activity diagram of ALO algorithm, the sequence of the activities, beginning from the starting point until the finishing point of the activity. To explain the requirements of ALO algorithm in tuning parameters, the use case diagram is modeled the functionality of the system using actors and uses case as illustrated in Figure 4.

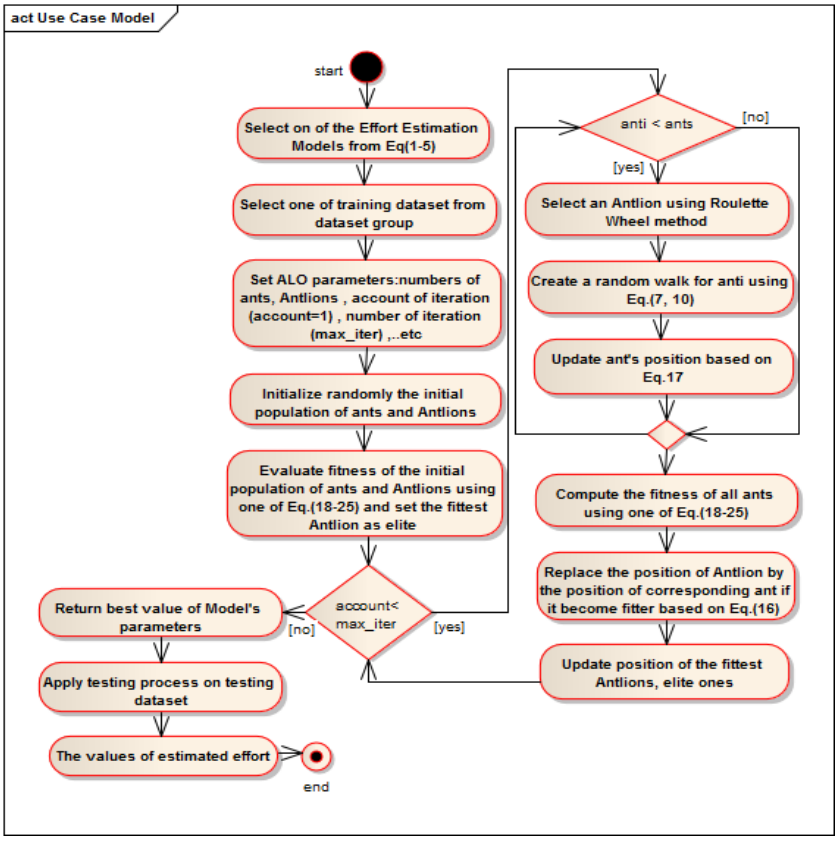


Figure 3. Activity diagram of ALO algorithm in tuning parameters

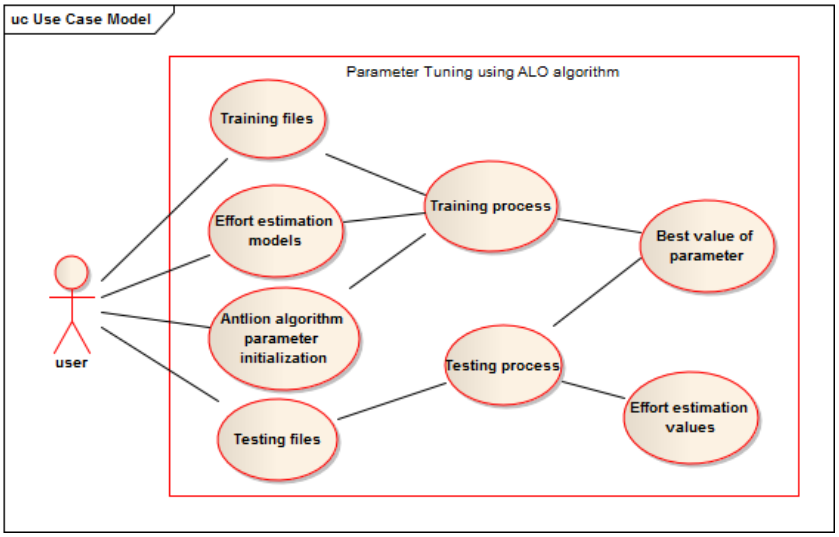


Figure 4. Use case diagram of ALO algorithm in tuning parameters

3. RESULTS AND ANALYSIS

After describing the ALO in the previous section, this section will demonstrate the datasets that are used in this study. This is done along with the evaluation criteria that are widely used to evaluate the quality of software effort estimation models and experiments. Results are analysed and compared with other methods.

3.1. Experimental datasets

The previously discussed algorithm is implemented using MATLAB 2017 in this work. Actual common datasets are used during the analysis; and are downloaded from the promise data repository, they are:

- Bailey and Basili [28], widely employed in many of the research studies, such as Sheta [20] and Uysal [21]. It consists of two independent variables: line of code (LOC) and methodology (ME), in addition to one dependent variable: measured effort (man-months).
- Dataset1 for the cocomo81 dataset covering 63 projects [29].
- Dataset2 for NASA dataset containing 60 projects [30].
- Dataset3 for NASA dataset with 93 project [31].
- Dataset4 for kemerer dataset having 15 projects [32].

Each software project has its actual cost available in the dataset; it is used in comparisons with estimated costs so as to determine the mean relative error for projects. We conducted three tests on the aforementioned data sets. In the first and second tests we used Bailey and Basili dataset but on different models, in first test we optimize parameters for three models: basic COCOMO model, Sheta's Model (named Model I) and Sheta's Model 2 (named Model II).In second test we optimize the parameters of four models Sheta's Model 1 (Model I), Sheta's Model 2 (Model II), Uysal's Model 1 (Model III) and Uysal's Model 2 (Model IV).The other four datasets were used in the third test to estimate the parameters of the basic COCOMO model.

3.2. Measures for evaluation

A number of evaluation criteria is used to evaluate the efficiency of the developed model, they are:

- Variance-Accounted-For (VAF) given in (18) [1]

$$\mathrm{VAF} = \left[1 - \frac{\mathrm{var}(\mathrm{y}-\mathrm{y}')}{\mathrm{var}\,(\mathrm{y})}\right] * 100\% \quad (18)$$

- Mean magnitude of relative error (MMRE) described in (19) [1]

$$MMRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \acute{y}_i|}{y_i} \quad (19)$$

- The prediction at level N (PRED(N)) stated in (20) [9]

$$\mathrm{PRED(L)} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}\begin{cases} 1 \text{ if MRE} \le \mathrm{L} \\ 0 \text{ otherwise} \end{cases} * 100 \quad (20)$$

- Mean absolute error (MAE) as in (21) [9]

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|E - \hat{E}| \quad (21)$$

- Mean squares error (MSE) as in (22) [7]

$$\mathrm{MSE} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{y} - \mathrm{y}')^2 \quad (22)$$

- The correlation coefficient (R2) as in (23) [7]

$$R^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 - \sum_{i=1}^{n}(y_i - \acute{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \quad (23)$$

- Root mean square error-RMSE as in (24) [33]

$$\mathrm{RMSE} = \sqrt{\frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}}(\mathrm{E_i} - \hat{\mathrm{E}}_\mathrm{i})^2} \quad (24)$$

- Median magnitude of relative error (MdMRE**)** as in (25) [33]

$$MdMRE = median(\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \acute{y}_i|}{y_i}) \quad (25)$$

3.3. Test1

The dataset of Bailey and Basili is employed along with ALO to optimize parameters for three models: basic COCOMO model, Sheta's Model (named Model I) and Sheta's Model 2 (named Model II). The range of parameters used here are as presented in [20] and [7]. For comparison purposes, the population

size or number of antlion agents is set to 100 and the iteration number is set to 500 according to that of [7], for the same purpose, the mean absolute error (MAE) criteria is employed as the objective function.

The data are divided into training and testing data; testing data is used to evaluate the optimized models by means of the following evaluation metrics VAF, MSE, MAE, MMRE, RMSE and R2. Results are compared with firefly algorithm (FA), genetic algorithm (GA), and particle swarm optimization (PSO). The ALO is applied to obtain the optimizing parameters for the three models, results are shown in Table 1. Tables 2, 3, and 4 show the evaluation results for testing the three models along with the comparison between ALO, FA, GA and PSO. From the results is seen that ALO exceeds FA, GA and PSO in the optimization of all models with all the evaluation metrics.

Table 1. Optimizing parameters using ALO

Models	Value of parameters
Basic COCOMO Model	A=1.9391, B=0.90839
Model I	A=1.9174, B=0.90948, C=0.024546
Model II	A=0.77671, B=1.101, C=-0.11225, D=8.7914

Table 2. Comparison for basic COCOMO model

	ALO	FA	GA	PSO
VAF	99.18%	98.16%	97.97%	97.98%
MSE	27.62	59.14	63.96	63.68
MAE	3.74	5.65	6.06	6.04
MMRE	0.06	0.11	0.13	0.12
RMSE	5.25	7.67	8.00	7.98
R2	0.9964	0.9781	0.9763	0.9765

Table 3. Comparison for Model I

	ALO	FA	GA	PSO
VAF	99.14%	98.62%	97.97%	98.52%
MSE	26.93	47.74	98.17	60.07
MAE	3.76	5.56	7.70	5.63
MMRE	0.07	0.24	0.29	0.23
RMSE	5.19	6.82	9.39	7.72
R2	0.9963	0.9823	0.9637	0.9778

Table 4. Comparison for Model II

	ALO	FA	GA	PSO
VAF	99.39%	98.63%	97.60%	98.70%
MSE	21.85	45.02	114.79	52.85
MAE	3.45	5.57	7.83	5.29
MMRE	0.10	0.24	0.27	0.21
RMSE	4.67	6.62	9.86	7.19
R2	0.9976	0.9833	0.9575	0.9805

3.4. Test2

In this section, the parameters of four models Sheta's Model 1 (Model I), Sheta's Model 2 (Model II), Uysal's Model1 (Model III) and Uysal's Model 2 (Model IV) are optimized. Bailey and Basili dataset is also used in this test, ALO parameter setting is set identical to [9], the iteration number is set to 100 and the population size is set to 10. The optimized parameters are shown in Table 5 using ALO and the four models.

Table 5. Best values of model's parameters using ALO

Models	Best values of parameters
Model I	A=1.0558, B=1.0378, C=0.097135
Model II	A=1.01356, B= 1.06415, C= -0.5, D=16.6215
Model III	A=1.172, B=1.0201, C= -0.11415, D= 1.2049, E=8.8288
Model IV	A=1.0442, B=1.0484, C=-0.2539, D=1.1643, E=2.3631, F=0.13983, G=6.9971

Initially, the accuracy of the models is assessed using MMRE, MdMRE, and PRED (25) criteria. Table 6 displays the gained results of the models using ALO in a comparison with GA for (Model I and Model II) and SA for (Model III and Model IV) that is given in [9] using directed artificial bee colony algorithm (DABCA) for the same four models. Results presented in Table 6 indicate that for Model I, the MMRE and MdMRE values for ALO are better than those of the others algorithms, PRED(25) is the same for all. This conclusion can also be drawn about Model II and Model III, with PRED(25) being better in both models. In Model IV however, ALO has the same good results in terms of MMRE and MdMRE but the PRED(25) value of DABCA is better than that of ALO.

Table 7 displays the predicted and the actual effort values for Model I and Model II using ALO, DABCA, and GA for (18) projects. For Model I, ALO found better estimates in term of the actual for (10)

projects, while DABCA found (6), and GA found only (2). As for Model II, ALO achieved (11) better estimated projects, DABCA achieved (4), and GA achieved (3) only. Table 8 gives the same values for Model III and Model IV using ALO, DABCA, and SA for the same (18) projects. In Model III, ALO was capable of finding best estimates for (10) projects, DABCA obtained (5), and SA found (3). On the other hand, in Model IV, ALO succeeded in finding (8) best estimated projects, DABCA found (6), and SA found just (4).

Table 6. Results based on MMRE, MdMRE and PRED(25)

Model	PRED(25)	MdMRE(%)	MMRE(%)
Model I (GA)	61.11	14.5	23.79
Model I (DABCA)	61.11	14.86	26.03
Model I (ALO)	61.11	13.39	18.43
Model II (GA)	38.89	49.27	63.64
Model II (DABCA)	77.78	11.48	17.13
Model II (ALO)	77.77	7.4	13.60
Model III (SA)	77.78	8.2	20.04
Model III (DABCA)	83.33	8.65	14.20
Model III (ALO)	77.77	5.82	13.55
Model IV (SA)	77.78	8.63	18.80
Model IV (DABCA)	83.33	7.07	13.21
Model IV (ALO)	83.33	5.37	13.36

Table 7. Measured data and predicted values for Model I and Model II

Proj	Model I			Model II			Actual Cost
	ALO	DABCA	GA	ALO	DABCA	GA	
1	15.7999	122.617	124.8585	123.6537	130.6186	134.0202	115.8
2	58.3195	75.9229	74.8467	66.5000	71.8103	84.1616	96
3	58.6023	76.6194	75.4852	67.4139	72.7508	85.0112	79
4	68.8641	86.1377	85.4349	78.0100	83.9645	94.9828	90.8
5	40.7873	51.0323	50.5815	38.4189	41.9945	56.658	39.6
6	125.1979	98.4043	99.0504	134.6901	98.378	107.2609	98.4
7	17.4059	24.8788	24.148	18.9000	18.9008	32.6461	18.9
8	15.4182	17.992	18.0105	11.9964	11.3608	25.0755	10.3
9	28.5000	38.002	37.2724	27.6527	29.6205	44.3086	28.5
10	5.9414	3.8676	4.5849	7.0000	3.7394	14.4563	7
11	6.5270	9.0108	8.9384	11.7889	9.0007	19.9759	9
12	11.9109	13.4767	13.5926	10.1406	8.6707	21.5763	7.3
13	5.0000	0.303	1.51	4.8537	1.1195	11.2703	5
14	8.4268	7.8061	8.2544	7.7404	5.2715	17.0887	8.4
15	101.2579	108.7481	110.5249	104.5251	111.4279	118.0378	98.7
16	13.7817	18.648	18.2559	14.4956	13.6236	26.8312	15.6
17	17.1413	24.0082	23.369	18.0192	17.9404	31.6864	23.9
18	129.9849	132.1635	135.4825	136.9700	143.8064	144.4587	138.3

3.5. Test3

In this test, ALO is used to estimate the parameters of the basic COCOMO model using four large datasets (dataset1, dataset2, dataset3 and dataset4), with the mean relative error (MRE) being the objective function. Results are compared with hybrid bat inspired gravitational search algorithm method called (BATGSA), the improved BAT (IBAT), and the BAT algorithms as presented in [12].

Results for using these four datasets are as follows:

- Dataset1 (63 project): ALO was used to estimate the values of parameters of basic COCOMO model, best values are a=2.3947, b=0.94614. Table 9 compares among all algorithms, based on MAE. Results show that the value of MAE of ALO is worse than the values of the other algorithms. As for the number of best estimated efforts, ALO achieved better results than BAT and IBAT, but BATGSA achieved the best estimates of (39), as shown in Table 9.
- Dataset2 (60 project): best values of parameters obtained using ALO are a=3.678, b=1.0474. Table 10 shows that the value of MAE for ALO is very close to that of BATGSA algorithm which is the best among the others. As for estimated efforts, ALO found the highest number of best estimates of (36) as presented in Table 10.
- Dataset3 (93 project): ALO obtained best values of parameters as a=2.1657, b=1.082. Table 11 illustrates the MAE and comparison among all algorithms. Results indicate that the value of MAE of ALO is worse than the value of MAE for the other algorithms. The values of estimated efforts show that ALO was better than BAT and IBAT, but BATGSA found the best estimates of (64) as shown in Table 11.

– Dataset4 (15 project): the parameters of basic COCOMO model are estimated using ALO, best values of parameters are a=8.3445, b=0.5187. Table 12 resembles a comparison among all algorithms. Results designate that the value of MAE for ALO is better than the values of MAE of all other algorithms. Results of the estimated efforts illustrate that ALO succeeded in achieving better estimates in all 15 project than other methods in comparison with actual effort.

Table 8. Measured data and predicted values for Model III and Model IV

Proj	Model III			Model IV			Actual Cost
	ALO	DABCA	SA	ALO	DABCA	SA	
1	117.6553	127.1478	124.794	119.4590	125.3071	124.3563	115.8
2	63.0827	78.2194	81.6608	64.3798	77.743	81.6143	96
3	63.7228	79.4325	83.1941	65.1364	79.1179	83.1781	79
4	73.8163	89.7282	92.8603	75.3859	89.2478	92.757	90.8
5	39.6000	39.5325	39.0238	38.2954	39.5424	39.6279	39.6
6	127.4898	98.3011	98.0132	129.8607	97.2828	97.8566	98.4
7	18.8307	23.3181	23.8838	18.9000	21.3727	23.8446	18.9
8	13.7344	9.5814	7.7948	12.5379	8.5967	8.2993	10.3
9	28.4731	30.8556	30.8864	27.7490	29.6235	31.0829	28.5
10	6.7589	6.96	5.3694	7.0000	6.441	5.7918	7
11	9.9296	15.4219	16.4089	11.0320	14.9203	16.7359	9
12	11.2023	9.223	7.6178	10.5595	7.75	7.8978	7.3
13	5.0000	2.8414	0.2631	4.9585	3.3478	0.9986	5
14	8.2808	6.9379	5.1496	8.0212	5.6857	5.4465	8.4
15	101.0944	105.0602	102.5719	101.4476	104.7675	102.7935	98.7
16	14.6714	17.2108	17.0202	14.6292	15.2793	17.0407	15.6
17	18.1841	21.7401	21.9803	18.1081	19.8065	21.9698	23.9
18	130.4215	135.5447	131.2398	132.1492	133.8053	130.9554	138.3

Table 9. Comparing algorithms using MAE and best estimates (dataset1)

	ALO	BATGSA	IBAT	BAT
MAE	551.2843	433.215	437.537	444.405
No. of Best Estimates from 63 Project	14	39	9	1

Table 10. Comparing algorithms using MAE and best estimates (dataset2)

	ALO	BATGSA	IBAT	BAT
MAE	127.6691	127.530	132.486	137.401
No. of Best Estimates from 60 Project	36	10	4	10

As the results show, ALO Algorithm was able to achieve better estimates than the other algorithms in Test1 and Test2. As for Test3, ALO succeeded in accomplishing better result than those of BAT and IBAT in all comparisons using all four datasets in terms of MAE and the number of best estimated projects for each dataset. In the comparison with BATGSA, ALO was able to achieve better results for 2 datasets (dataset2 and dataset4) out of the four investigated datasets also in terms of MAE and the number of best estimated projects for each dataset.

Table 11. Comparing algorithms using MAE and best estimates (dataset3)

	ALO	BATGSA	IBAT	BAT
MAE	378.7037	355.386	356.143	365.164
No. of Best Estimates from 93 Project	18	64	5	6

Table 12. Comparing algorithms using MAE and best estimates (dataset4)

	ALO	BATGSA	IBAT	BAT
MAE	118.7845	2398.141	4564.934	5528.158
No. of Best Estimates from 15 Project	15	0	0	0

4. CONCLUSION

The parameters of the considered mathematical models are optimized in this work using the ALO algorithm. ALO is compared with various algorithms in a three-test approach to assess its efficiency. In Test1, the optimized models were evaluated using various evaluation metrics, namely: VAF, MSE, MAE, MMRE, RMSE and R2. Results of comparisons with FA, GA, and PSO showed that ALO succeeded in achieving the best estimates using Bailey and Basili dataset.

ALO was compared with DABCA, GA, and SA in Test2, the criteria used in this test were: MMRE, MdMRE, and PRED (25). Results also indicated the superiority of ALO over other methods for Bailey and Basili dataset. As for Test3, ALO was capable of accomplishing better result than those of BAT and IBAT in all comparisons using all four datasets in terms of MAE and the number of best estimated projects for each dataset. In the comparing with BATGSA, ALO was able to achieve better results for 2 datasets (dataset2 and dataset4) out of the four investigated datasets also in terms of MAE and the number of best estimated projects for each dataset.

The future work focuses on employing ALO algorithm in tuning parameters for other effort estimation models. Plants optimization studies have shown in recent years that plants possess intelligent behaviors. One of the plant intelligent algorithms can be used in parameters tuning. In addition, new models can be suggested for estimating the effort for projects that are influenced by the size and type of the datasets.

ACKNOWLEDGEMENTS

The authors are very grateful to the University of Mosul/ College of Computer Science and Mathematics for their provided facilities, which helped to improve the quality of this work.

REFERENCES

[1] A. Abu-Srhan, A. Sleit and A. Sharieh, "Parameters Estimation of the COCOMO Model Using Hybrid Algorithm of Genetic Algorithm and Cuckoo Search Algorithm," *Proceedings of the New Trends in Information Technology*, University of Jordan, Amman, pp. 67-73, 2017.

[2] A. Hussain, M. Mohanapriya and S. Rajalakshmi, "Software Effort Estimation Using Modified Fuzzy C Means Clustering and Hybrid ABC-MCS Optimization in Neural Network," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 251-263, 2018.

[3] S. Aljahdali and A. F. Sheta, "Software Effort Estimation by Tuning COCOMO Model Parameters Using Differential Evolution," *ACS/IEEE International Conference on Computer Systems and Applications - AICCSA* 2010, Hammamet, Tunisia, pp. 1-6, 2010.

[4] B. K. Singh and A. K. Misra, "Software Effort Estimation by Genetic Algorithm Tuned Parameters of Modified Constructive Cost Model for NASA Software Projects," *International Journal of Computer Applications*, vol. 59, no. 9, pp. 22-26, 2012.

[5] A. Kundu and V. Sethi, "Parameter Estimation of COCOMO II using Simulated Annealing," *International Journal of Science and Research (IJSR)*, vol. 3, no. 8, pp. 530-534, 2012.

[6] A. Dhiman and C. Diwaker, "Optimization of COCOMO II Effort Estimation using Genetic Algorithm," *American International Journal of Research in Science, Technology, Engineering & Mathematics*, vol. 3, no. 2, pp. 208-212, 2013.

[7] 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*, vol. 8, no. 3, pp. 133-142, 2015.

[8] N. Gupta and K. Sharma, "Optimizing Intermediate COCOMO Model Using BAT Algorithm," *2nd International Conference on Computing for Sustainable Global Development (INDIACom*), *New Delhi*, 2015, pp. 1649-1653.

[9] T. T. Khuat and M. L. Hanh, "Optimizing Parameters of Software Effort Estimation Models using Directed Artificial Bee Colony Algorithm," *Informatica*, vol. 40, pp. 427-436, 2016.

[10] V. K. Bardsiri and M. Dorosti, "An Improved COCOMO based Model to Estimate the Effort of Software Projects," *Journal of Advances in Computer Engineering and Technology*, vol. 2, no. 2, 2016.

[11] S. Girotra, and K. Sharma, "Tuning of Software Cost Drivers using Bat Algorithm," *3rd International Conference on Computing for Sustainable Global Development (INDIACom). IEEE*. 16-18 March 2016. New Delhi, India, 2016, pp. 1051-1056.

[12] D. Nandal and O. P. Sangwan, "Software Cost Estimation by Optimizing COCOMO Model Using Hybrid BATGSA Algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 11, no. 4, pp. 250-263, 2018.

[13] E. Khatibi and V. K. Bardsiri, "An Improved Algorithmic Method for software Development Effort Estimation," *Journal of Advances in Computer Research*, vol. 9, no.1, pp. 40-49, 2018.

[14] S. A. A. Dizaj and F. S. Gharehchopogh, "A New Approach in Software Cost Estimation by Improving Genetic Algorithm with Bat Algorithm," *Journal of Computer & Robotics.* vol. 11, no. 2, pp. 17-30,2018.

[15] V. Venkataiah, R. Mohanty and M. Nagaratna, "Application of Hybrid Techniques to Forecasting Accurate Software Cost Estimation," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 7, no. 6S, pp. 408-412, 2019.

[16] T. Špoljarić and I. Pavić, "Performance Analysis of an Ant Lion Optimizer in Tuning Generators' Excitation Controls in Multi Machine Power System," *41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija*, 2018, pp. 1040-1045.
[17] B. W. Boehm, "Software Engineering Economics," *IEEE Transactions on Software Engineering*, vol. 10, no. 1, pp. 4-21, 1984.
[18] N. Sharma, A. Sinhal and B. Verma, "Software Assessment Parameter Optimization using Genetic Algorithm," *International Journal of Computer Applications*, vol. 72, no. 7, pp. 8-13, 2013.
[19] I. Z. Quba, "Software Projects Estimation using Neural Networks," *M.Sc. Thesis. College of Computer Sciences & Mathematics, University of Mosul*. (In Arabic), 2012.
[20] A. F. Sheta, "Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects," *Journal of Computer Science*, DOI: 10.3844/jcssp.2006.118.123, vol. 2, no. 2, pp. 118-123, 2006.
[21] M. Uysal, "Estimation of the effort component of the software projects using simulated annealing algorithm," *World Academy of Science, Engineering and Technology*, vol. 41, pp. 258–261, 2008.
[22] M. Petrović, *et al.*, "The Ant Lion Optimization Algorithm for Flexible Process Planning," *Journal of Production Engineering*, vol. 18, no. 2, pp. 65-68, 2015.
[23] S. Mirjalili, "The Ant Lion Optimizer," *Advances in Engineering Software*, vol. 83, pp. 80-98, 2015.
[24] R. Satheeshkumar and R. Shivakumar, "Ant Lion Optimization Approach for Load Frequency Control of Multi-Area Interconnected Power Systems," *Journal of Scientific Research Publishing*, vol. 7, no. 9, pp. 2357-2383, 2016.
[25] S. Talatahari, "Optimum Design of Skeletal Structures Using Ant Lion Optimizer," *International Journal of Optimization in Civil Engineering*, vol. 6, no. 1, pp. 13-25, 2016.
[26] M. Nischal and S. Mehta, "Optimal Load Dispatch Using Ant Lion Optimization," *Int. Journal of Engineering Research and Applications*, ISSN: 2248-9622, vol. 5, no. 8, (Part - 2) August, pp. 10-19, 2015.
[27] E. S. Ali, S. M. Abd Elazima and A.Y. Abdelaziz, "Ant Lion Optimization Algorithm for optimal location and sizing of renewable distributed generations," *Renewable Energy*, vol. 101, pp. 1311 -1324, 2017.
[28] J. W. Bailey and V. R. Basili, "A Meta Model for Software Development Resource Expenditure," *Proceedings of the 5th International Conference on Software Engineering*, 1981, pp. 107-116.
[29] http://promise.site.uottawa.ca/SERepository/datasets/cocomo81.arff.
[30] http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_v1.arff.
[31] http://promise.site.uottawa.ca/SERepository/datasets/cocomonasa_2.arff.
[32] C. F. Kemerer, "An Empirical Validation of Software Cost Estimation Models," *Communications of the Association for Computing Machinery*. vol. 30, no. 5, pp. 416–429, 1987.
[33] T. Urbanek, Z. Prokopova, R. Silhavy and V. Vesela, "Prediction Accuracy Measurements as a Fitness Function for Software Effort Estimation," *Springer Plus*, vol. 4, no. 778, pp. 1-17, 2015.