Received: March 26, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Software Cost Estimation by Optimizing COCOMO Model Using Hybrid BATGSA Algorithm

Deepak Nandal[1]* **Om Prakash Sangwan[1]**

[1]*Department of Computer Science and Engineering,*
Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India
* Corresponding author's Email: dr.deepaknandalgju@gmail.com

Abstract: This paper estimates the effort for software by optimizing the COnstructive COst MOdel (COCOMO) model parameters using hybrid BATGSA (Bat inspired Gravitational Search Algorithm) algorithm. The performance of the COCOMO model completely depends upon its parameters which can be optimized by using meta-heuristic algorithms. This paper uses hybrid BATGSA algorithm which hybrids the improved bat algorithm with the gravitation search algorithm (GSA) to optimize the COCOMO model. The bat algorithm demonstrates the hunting and routing behavior of the bat which is improved by using a random walk in the exploration phase. The exploration phase is further improved by using GSA as gravitation force affects the velocity of the bat. The algorithm has been analyzed on four NASA datasets downloaded from promise repository. The comparison of the algorithm has been made with existing three states of art techniques i.e. COCOMO model, BAT algorithm, Improved BAT(IBAT) algorithm by using normalized error as a parameter. The reduction in error ranges from 2% to 10% on different dataset as compared other state of art algorithms proves the significance of proposed algorithm.

Keywords: NASA project dataset, COCOMO, BAT, GSA, Optimization.

1. Introduction

Software cost estimation is a critical process as it is required to estimate the cost of the project at the initial stage of the project. The cost estimation is required to compute the budget and the resources required for the project [1, 2]. The cost of the project depends upon the efforts done which include the number of reviews, efficiency during implementation and pre-development processing, etc [3]. The Constructive Cost Model (COCOMO) is a model to determine the cost of the software. This model uses a basic regression formula for estimating cost, effort, and schedule for software projects based on features of the project. The model calculates the efforts as $E=a \times (KLOC)b$, here KLOC shows the kilo line of code and a, b are the constants which depends upon the type of software [4]. This model can be optimized by using meta-heuristic algorithm [5]. Different authors have applied the various meta-heuristic algorithm to improve the performance of the COCOMO model. The author of [6] has applied a genetic algorithm to optimize the parameters of COCOMO model while the author of [1] has applied differential evolution algorithm for the same. The author of [7] has applied BAT algorithm to improve the performance of cost estimation by COCOMO. The author of [7] shows the reduction in error by optimizing the COCOMO model using BAT algorithm. The BAT algorithm shows the better result as compared other existing state of the art techniques but BAT algorithm search locally in the exploration phase which opens scope for the improvement. This paper optimizes the estimation using COCOMO model by using hybrid BATGSA algorithm. The hybrid BATGSA algorithm has better convergence towards the global optima as it has improved exploration phase as compared to BAT algorithm, resulting in better optimization of COCOMO model. This paper further has been classified in five sections. The next section i.e. section 2 describes the improved BAT algorithm which is followed by the description of Gravitation search algorithm (GSA) in section 3. The Hybrid BATGSA algorithm has been described in section 4 while the results and corresponding discussion is done in section 5.

2. Improved BAT algorithm for cost estimation

Bat algorithm performs well for the various applications [8, 9]. This algorithm automatically switches from the exploration phase i.e. global search, to the exploitation phase i.e. local search. The main limitation of the algorithm is that it mainly searches locally even in the exploration phase [7, 9]. In the exploration phase change in the frequency is the parameter for the global search which restricts the search locally mainly. This behavior has been changed by introducing the random move nature to the bat resultant velocity updation is given by Eq. (1). Suppose there are n bats with position vector P[1 : n] and velocity vector V[1 : n] i.e. P_a^t, V_a^t denotes the position and velocity of the a[th] bat at t[th] time. Moreover, there is a frequency vector f[1 : n] that denotes the frequency of each bat which lies between f_{min} and f_{max} The velocity of each bat is adjusted according to frequency using the equation

$$V_a^t = \begin{cases} V_a^{t-1} + (P_a^{t-1} - P^b) f_a & pr < 0.5 \\ V_a^{t-1} + (P_a^{t-1} - P^r) \mathrm{X} pr & else \end{cases} \tag{1}$$

Here pr is the probability which is generated randomly by using a random function and P^r is position at any random time stamp. This updation strength the exploration phase and improves the global search. Correspondingly position update can be given by Eq. (2) to improve the impact of velocity on the position.

$$P_a^t = \begin{cases} P_a^{t-1} + V_a^t & pr < 0.5 \\ P_a^{t-1} + V_a^t f_a & else \end{cases} \tag{2}$$

The Eq. (2) updates the position of the bat according to the velocity update. The local search phase of the algorithm is powerful enough to exploit the search space. The improved bat algorithm is as follow:

IBAT Algorithm (P, V, f, F, emr^0, L):

Here, P, V, f are the position, velocity and frequency vector for n bats respectively. The F is the fitness function to evaluate the fitness of the solution generated by the corresponding position. The L is the loudness vector and emr^0 is initial emission rate.

1. Initiate iteration=1
2. $P^b=P_1$
3. For i=2:n
    a. If $F(P_i) > F(P^b)$
    b. $P^b=P_i$
    c. End if

   End
4. While iteration<max_iteration
    a. $f_{iteration} = f_{\min} + (f_{\max} - f_{\min})\alpha$
    b. pr=rand
    c. r=rand(1,n)
    d. $V_{1:n}^{iteration} = \begin{cases} V_{1:n}^{iteration-1} + (P_{1:n}^{iteration-1} - P^b) f_a & pr < 0.5 \\ V_{1:n}^{iteration-1} + (P_{1:n}^{iteration-1} - P^r) \mathrm{X} pr & else \end{cases}$
    e. $P_{1:n}^{iteration} = \begin{cases} P_{1:n}^{iteration-1} + V_{1:n}^{iteration} & pr < 0.5 \\ P_{1:n}^{iteration-1} + V_{1:n}^{iteration} f_{1:n} & else \end{cases}$
    f. If rand > $emr^{iteration-1}$

       $P_{1:n}^{iteration} = P_{1:n}^{iteration-1} + \varepsilon L^{iteration}$

       End if
    g. If F($P_{1:n}^{iteration}$) > F($P_{1:n}^{iteration-1}$)

       $P_{1:n}^{iteration} = P_{1:n}^{iteration-1}$

       Else

       $L_{1:n}^{iteration} = L_{1:n}^{iteration-1}\beta$

       $emr_{1:n}^{iteration} = emr_{1:n}^{0}(1 - e^{-\lambda(iteration-1)})$

       End if
    h. For i=1:n
        i. If $F(P_i) > F(P^b)$
        ii. $P^b=P_i$
        iii. End if

       End
    i. iteration++

   End while

The above algorithm gives the process of improved bat algorithm [10, 11]. This algorithm has been applied to the cost estimation of software by using the objective function given by Eq. (3)

$$F(P) = \left(\sum_{i=1}^{n} (actual_i - predicted_i) / actual_i\right) \Big/ n \quad (3)$$

which calculates the mean relative error which depends upon the actual and the predicted estimation. The performance of the algorithm can be improved discussed in next section.

3. Gravitational search algorithm

The gravitational search algorithm uses the law of gravity given by Newton [12]. According to Newton every particle attracts another particle with a gravitational force that can be given by Eq. (4).

$$F = G\frac{M_1 M_2}{D^2} \quad (4)$$

Here, G is the gravitational constant, D is the distance between two objects having masses M_1 and M_2 respectively. The details of GSA are as follow.
Suppose we have n objects in d dimensions then position matrix for objects can be given by Eq. (5).

$$O = \begin{bmatrix} O_1^1 & O_1^2 & L & O_1^d \\ O_2^1 & O_2^2 & L & O_2^d \\ M & M & M & M \\ O_n^1 & O_n^2 & L & O_n^d \end{bmatrix} \quad (5)$$

Here, O_i^j represents the position of i^{th} object in the j^{th} dimension. At a particular time stamp say t the force acting on object p from object q can be given by Eq. (6).

$$F_{pq}^{1:d}(t) = G(t)\frac{M_p^a(t) M_q^p(t)}{D_{pq}(t)+\varepsilon}\left(O_i^{i:d}(t) - O_j^{1:d}(t)\right) \quad (6)$$

Here $M_p^a(t), M_q^p(t)$ are active and passive gravitational masses of object p and q respectively. ε is a small constant with value 2^{-52}, $D_{pq}(t)$ is the distance between object p and object q which is given by Eq. (7).

$$D_{pq}(t) = \sqrt{\sum_{i=1}^{d} (O_p^i - O_q^i)^2} \quad (7)$$

Overall the total force acting on any object p can be given by Eq. (8)

$$F_p^{1:d}(t) = \sum_{i=1, i\neq p}^{n} F_{pi}^{1:d} Xrand_i \quad (8)$$

So the acceleration is given by Eq. (9).

$$a_p^{1:d}(t) = \frac{F_p^{1:d}(t)}{O_p(t)} \quad (9)$$

Here, $O_p(t)$ is the inertia mass of object p.

This acceleration is used to update the velocity of the object which affects the position of an object demonstrated by Eqs. (10) and (11).

$$v_p^{1:d}(t+1) = v_p^{1:d}(t) Xrand + a_p^{1:d}(t) \quad (10)$$

and

$$O_p^{1:d}(t+1) = O_p^{1:d}(t) + v_p^{1:d}(t+1) \quad (11)$$

The whole process is repeated for the updated position for the given number of iterations or until fulfilling the stopping criteria. The overall GSA algorithm can be given as follow.

GSA (O, M, Fit)

The algorithm uses O as the object position matrices with M as the mass metric containing mass of each object. Fit is the fitness function.

1. Initiate iteration=1
2. $O = \begin{bmatrix} O_1^1 & O_1^2 & L & O_1^d \\ O_2^1 & O_2^2 & L & O_2^d \\ M & M & M & M \\ O_n^1 & O_n^2 & L & O_n^d \end{bmatrix}$
3. While iteration< max_iteration
   a. $Fit(t) = \begin{bmatrix} Fit(O_1^1(t)) & Fit(O_1^2(t)) & L & Fit(O_1^d(t)) \\ Fit(O_2^1(t)) & Fit(O_2^2(t)) & L & Fit(O_2^d(t)) \\ M & M & M & M \\ Fit(O_n^1(t)) & Fit(O_n^2(t)) & L & Fit(O_n^d(t)) \end{bmatrix}$
   b. $D_{pq}(t) = \sqrt{\sum_{i=1}^{d} (O_p^i - O_q^i)^2}$
   c. $F_{pq}^{1:d}(t) = G(t)\frac{M_p^a(t) M_q^p(t)}{D_{pq}(t)+\varepsilon}\left(O_i^{i:d}(t) - O_j^{1:d}(t)\right)$
   d. $F_p^{1:d}(t) = \sum_{i=1, i\neq p}^{n} F_{pi}^{1:d} Xrand_i$

e. $a_p^{1:d}(t) = \frac{F_p^{1:d}(t)}{O_p(t)}$

f. $v_p^{1:d}(t+1) = v_p^{1:d}(t)\mathrm{X}rand + a_p^{1:d}(t)$

g. $O_p^{1:d}(t+1) = O_p^{1:d}(t) + v_p^{1:d}(t+1)$

h. If Fit(t)<Fit(t+1)

$$O_p^{1:d}(t+1) = O_p^{1:d}(t)$$

end

End while

4. Return best(F)

The above algorithm performs the optimization. The fitness has been given by Eq. (3) to determine the software cost estimation. The improved bat algorithm is hybridized by using gravitational search algorithm discussed in next section.

4. Hybrid BATGSA algorithm

In the real-time the gravitational force as the impact the velocity of the BAT. in this work to improve the exploration phase of the algorithm the velocity factor has been affected by the gravitational force also. This improves the global search capability of the algorithm resulting better convergence for optimization. The concept can be easily understood by following algorithm:

BATGSA Algorithm (P, V, M, f, Fit, emr^0, L)

Here, P, V, M, f are the position, velocity, mass and frequency vector for n bats respectively. The Fit is the fitness function to evaluate the fitness of the solution generated by corresponding position. The L is the loudness vector and emr^0 is initial emission rate.

1. Initiate iteration=1

2. $P = \begin{bmatrix} P_1^1 & P_1^2 & L & P_1^d \\ P_2^1 & P_2^2 & L & P_2^3 \\ M & M & M & M \\ P_n^1 & P_n^2 & L & P_n^d \end{bmatrix}$

3. $Fit(iteration) = \begin{bmatrix} Fit(P_1^1(iteration)) & Fit(P_1^2(iteration)) & L & Fit(P_1^d(iteration)) \\ Fit(P_2^1(iteration)) & Fit(P_2^2(iteration)) & L & Fit(P_2^d(iteration)) \\ M & M & M & M \\ Fit(P_n^1(iteration)) & Fit(P_n^2(iteration)) & L & Fit(P_n^d(iteration)) \end{bmatrix}$

4. $P^b=P_1$

5. For i=2:n
   a. If $Fit(P_i) > Fit(P^b)$
   b. $P^b=P_i$
   c. End if

   End

6. While iteration<max_iteration
   a. $f_{iteration} = f_{\min} + (f_{\max} - f_{\min})\alpha$
   b. pr=rand
   c. r=rand(1,n)

d. $V_{1:n}^{iteration} = \begin{cases} V_{1:n}^{iteration-1} + (P_{1:n}^{iteration-1} - P^b) f_a & pr < 0.5 \\ V_{1:n}^{iteration-1} + (P_{1:n}^{iteration-1} - P^r) \mathrm{X} pr & else \end{cases}$

e. $P_{1:n}^{iteration} = \begin{cases} P_{1:n}^{iteration-1} + V_{1:n}^{iteration} & pr < 0.5 \\ P_{1:n}^{iteration-1} + V_{1:n}^{iteration} f_{1:n} & else \end{cases}$

f. If rand > $emr^{iteration-1}$

$$P_{1:n}^{iteration} = P_{1:n}^{iteration-1} + \varepsilon L^{iteration}$$

End if

g. For p=1:n

i. For q=1:n

$$D_{pq}(iteration-1) = \sqrt{\sum_{i=1}^{d} (P_p^i - P_q^i)^2}$$

$$F_{pq}^{1:d}(iteration-1) = G(iteration) \frac{M_p^a(t) M_q^p(t)}{D_{pq}(t) + \varepsilon} \left(P_p^{i:d}(iteration) - P_q^{1:d}(t) \right)$$

End

ii. $F_p^{1:d}(iteration-1) = \sum_{i=1, i \neq p}^{n} F_{pi}^{1:d} \mathrm{X} rand_i$

iii. $a_p^{1:d}(iteration-1) = \dfrac{F_p^{1:d}(iteration)}{P_p(iteration)}$

iv. $v_p^{1:d}(iteration) = v_p^{1:d}(iteration-1) \mathrm{X} rand + a_p^{1:d}(iteration-1)$

v. $P_p^{1:d}(iteration) = P_p^{1:d}(iteration-1) + v_p^{1:d}(iteration)$

End

h. If Fit($P_{1:n}^{iteration}$) > Fit($P_{1:n}^{iteration-1}$)

$$P_{1:n}^{iteration} = P_{1:n}^{iteration-1}$$

Else

$$L_{1:n}^{iteration} = L_{1:n}^{iteration-1} \beta$$

$$emr_{1:n}^{iteration} = emr_{1:n}^{0} (1 - e^{-\lambda(iteration-1)})$$

End if

i. For i=1:n

i. If $F(P_i) > F(P^b)$

ii. $P^b = P_i$

iii. End if

End

j. iteration++

End while

The above algorithm updates the position of the bat firstly on the basis of frequency of the sound then this position is updated on the basis of the gravitational force on bat for the global search while the local search still depends upon the loudness of the sound. This algorithm produces exploration as well as exploitation search so the optimization must converge towards the global optima. The objective function given in Eq. (3) has been used for the cost estimation. The implementation of algorithm and the result analysis has been discussed in next section.

5. Results and discussion

The algorithm discussed in previous section has been implemented using MATLAB. The analysis has been done on 4 datasets, 2 are the NASA datasets containing 60 and 93 projects respectively and other two are COCOMO81 and kemerer. All datasets has been downloaded from the promise data repository. The actual cost of each software project is available in the dataset which is compared with the estimated cost to calculate the mean relative error for each project. The comparison of the hybrid BATGSA algorithm is done with the COCOMO based cost estimation i.e. cost estimation without optimization [3] with the BAT [7] and improved BAT based optimization [11]. The remaining section of the paper uses Dataset1 for the cocomo81 dataset, Dataset2 for NASA dataset having 60 projects, Dataset3 for NASA dataset with 93 project and Dataset 4 for kemerer dataset. The software effort estimation for Dataset1 has been analyzed in Table1.

The effort estimation for each project in the Dataset1 has been shown in the table 1. The analysis Table 1 shows that the value of the effort converges towards the actual effort but for some project the estimation is producing higher error. This is because the algorithm reduces the mean absolute error in cost estimation by Eq. (8). The mean absolute error in the Dataset1 has been to 433.215 from 437.537(Improved BAT), 444.405 (Bat algorithm) which is reduced from 680.154 in COCOMO model. The normalized error in each project of Dataset1 has been analyzed in figure 1. The normalized error has been calculated by Eq. (12).

$$NE_i = abs(actual_i - predicted_i) / actual_i \quad (12)$$

Here, NE_i is the normalized error for i-th project which has been calculated by using actual and predicted effort of the project.

The Fig. 1 clearly shows that normalized error has been reduced by using hybrid BATGSA is more as compared to reduction by improving the bat algorithm and bat algorithm. This is due to the better optimization of COCOMO model by BATGSA algorithm which is due to better exploration phase of BATGSA algorithm.

The software cost/effort estimation for each undertaking in the Dataset2 has been appeared in the table 2. The mean absolute error in the Dataset2 has been to 127.530 from 132.486 in improved BAT and 137.401 (Bat calculation) which is decreased from 400.985 in COCOMO display. The standardized error in each project of Dataset2 has been investigated in Fig. 2 which is computed by Eq. (12).

The Fig. 2 unmistakably connotes that normalized error has been reduced by hybrid BATGSA calculation is higher than the reduction done by improved bat and bat calculation due to avoidance of local minima by the BATGSA algorithm.

The effort estimation for each project in the Dataset3 has been shown in the Table 3. The analysis table 3 shows that the value of the effort converges towards the actual effort but for some project the estimation is producing higher error. This is because the algorithm reduces the mean absolute error in cost estimation by Eq. (3). The mean absolute error in the Dataset1 has been to 355.386 from 356.143(improved BAT), 365.164 (Bat algorithm) which is reduced from 618.412 in COCOMO model.

The Fig. 3 clearly shows that normalized error has been reduced by using hybrid BATGSA is more as compared to reduction by improving the bat algorithm and bat algorithm. This is due to the better optimization of COCOMO model by BATGSA algorithm which is due to better exploration phase of BATGSA algorithm.

The software cost/effort estimation for each undertaking in the Dataset4 has been appeared in the table 4. The mean absolute error in the Dataset4 has been to 2398.141 from 4564.934 in improved BAT and from 5528.158 (Bat calculation) which is decreased from 6804.450 in COCOMO display. The standardized error in each project of Dataset4 has been investigated in figure 4 which is computed by Eq. (12).

The Fig. 4 unmistakably connotes that normalized error has been reduced by hybrid BATGSA calculation is higher than the reduction done by improved bat and bat calculation. This is due to the velocity variation done to explore the search space appropriately. This shows that the

Received: March 26, 2018

hybrid BATGSA algorithm improves the estimation of the software effort.

Table 1. Analysis of effort estimation on Dataset 1

Project Number	Hybrid BATGSA Optimization	IBAT Optimization	BAT Optimization	COCOMO	Actual
1	500.0036	454.2916	474.5296	3.134719	2040
2	1446.034	1362.665	1443.515	3	1600
3	594.5699	543.4377	568.9498	3	243
4	246.9118	218.9643	226.593	3.134719	240
5	56.58902	47.70643	48.41535	3	33
6	12.0693	9.64865	9.593832	3	43
7	22.16179	18.09085	18.13331	3	8
8	80.70177	68.86921	70.22168	3.270061	1075
9	114.0294	98.4732	100.8675	3	423
10	109.801	94.69864	96.95262	3	321
11	122.5342	106.0797	108.7624	3	218
12	144.0569	125.4077	128.8548	3.270061	201
13	93.0598	79.80509	81.52591	3	79
14	8.758434	6.925093	6.856592	3.782814	60
15	11.73347	9.371089	9.314369	3.782814	61
16	19.31758	15.69484	15.70312	3	40
17	10.73201	8.54502	8.483278	3	9
18	1595.32	1508.428	1600.005	3.270061	11400
19	6638.286	6591.914	7125.294	3	6600
20	1479.077	1394.886	1478.089	3.270061	6400
21	1222.389	1145.283	1210.54	3	2455
22	524.7245	477.5438	499.1363	3.782814	724
23	326.0589	291.9289	303.2134	3.782814	539
24	387.982	349.452	363.7963	3	453
25	148.4032	129.3233	132.9303	3.270061	523
26	192.5427	169.2959	174.6185	3.270061	387
27	31.28028	25.83853	26.01757	3.270061	88
28	44.8975	37.55026	37.99172	3	98
29	6.01046	4.69124	4.621749	3.782814	7.3
30	5.51178	4.289229	4.220852	3.782814	5.9
31	256.1027	227.4003	235.4368	3	1063
32	1988.874	1894.85	2015.761	3	702
33	165.9165	145.1403	149.4089	3	605
34	84.80089	72.49063	73.9627	3.782814	230
35	44.8975	37.55026	37.99172	3	82
36	52.66131	44.28563	44.90095	3.134719	55
37	246.9118	218.9643	226.593	3	47
38	52.66131	44.28563	44.90095	3	12
39	19.67088	15.99184	16.00411	3	8
40	8.758434	6.925093	6.856592	3	8

41	16.51591	13.34653	13.3259	3	6
42	181.3991	159.1714	164.0462	3.134719	45
43	108.1143	93.19433	95.39297	3.134719	83
44	116.5742	100.7472	103.2269	3.134719	87
45	135.4051	117.6252	120.7594	3.134719	106
46	307.2369	274.5156	284.9027	3.134719	126
47	84.80089	72.49063	73.9627	3.134719	36
48	2413.831	2315.072	2469.116	3.134719	1272
49	392.7899	353.9322	368.5205	3	156
50	88.92049	76.13621	77.73109	3	176
51	33.51367	27.74908	27.96687	3.134719	122
52	26.86333	22.07421	22.18244	3	41
53	16.51591	13.34653	13.3259	3.270061	14
54	13.42202	10.76931	10.72319	3	20
55	20.02484	16.28958	16.30592	3	18
56	101.3949	87.20966	89.19133	3.270061	958
57	60.54499	51.1601	51.96679	3.782814	237
58	93.0598	79.80509	81.52591	3	130
59	84.80089	72.49063	73.9627	3	70
60	21.44701	17.48765	17.5211	3.270061	57
61	105.5893	90.94392	93.06036	3	50
62	30.16963	24.89016	25.05064	3	38
63	33.51367	27.74908	27.96687	3	15

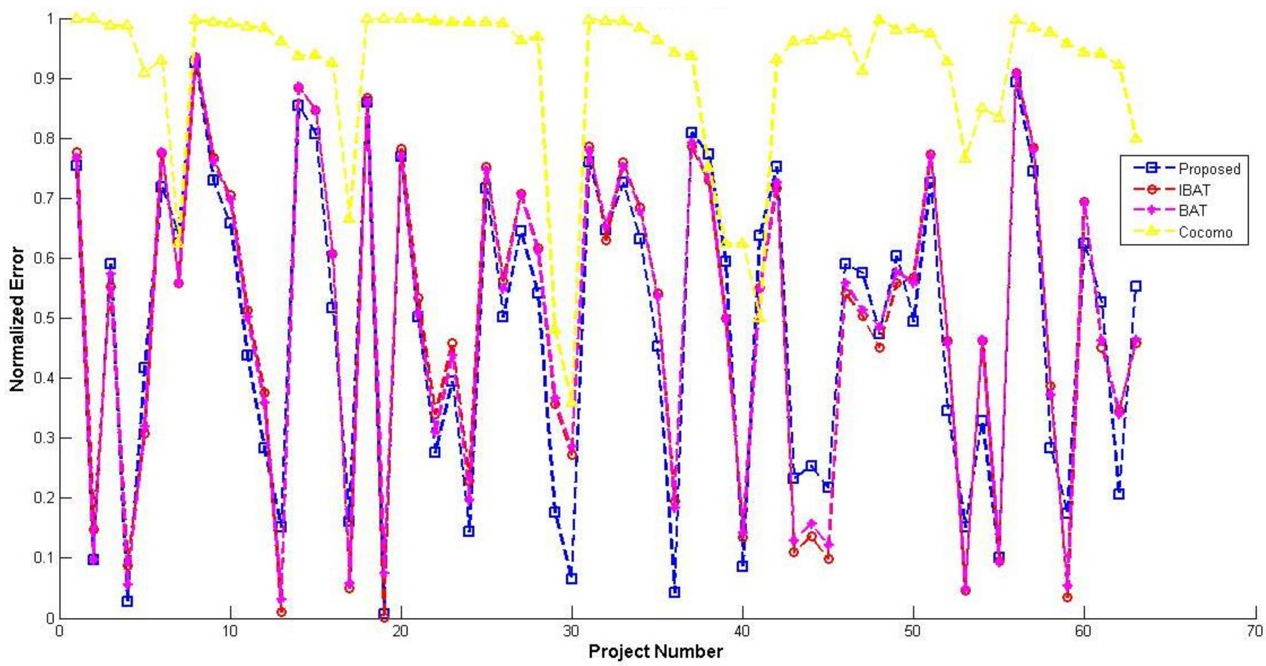

Figure. 1 Analysis of normalized error on Dataset 1

Table 2. Analysis of effort estimation on Dataset 2

Project Number	Hybrid BATGSA Optimization	IBAT Optimization	BAT Optimization	COCOMO	Actual
1	287.2043	265.0441	247.4643	3	278
2	1167.179	1070.171	1132.798	3	1181
3	872.9328	801.4547	826.5829	3	1248
4	523.2879	481.5761	474.4389	3	480
5	102.5464	95.08511	80.96048	3	120
6	63.37579	58.8953	48.03242	3	60
7	270.6567	249.8417	232.0345	3	300
8	13.85218	12.96357	9.227148	3	18
9	29.59828	27.60263	21.0285	3	50
10	42.18202	39.27361	30.88377	3	60
11	49.45893	46.01497	36.70414	6.520409	114
12	16.90391	15.80499	11.45188	6.520409	42
13	38.61608	35.96821	28.06175	6.520409	60
14	21.65029	20.2197	14.9788	6.520409	42
15	387.5022	357.109	342.4826	6.520409	450
16	45.79719	42.62334	33.76551	6.520409	90
17	138.6684	128.3999	112.3196	6.520409	210
18	28.24655	26.34772	19.98866	6.520409	48
19	775.5962	712.4771	727.0806	3	815
20	185.4654	171.5012	153.979	3	239
21	115.5157	107.0519	92.12647	3	170
22	37.90906	35.31268	27.50479	3	62
23	47.25644	43.97509	34.93428	3	70
24	50.56617	47.0403	37.59643	3	82
25	127.8651	118.441	102.8585	3	192
26	87.81213	81.48128	68.42088	3	117.6
27	82.58465	76.65239	64.01337	3	117.6
28	20.68617	19.32334	14.25653	3	31.2
29	27.23952	25.41264	19.21672	3	25.2
30	4.647672	4.371511	2.821779	3	8.4
31	8.082867	7.58319	5.14345	3	10.8
32	22.29692	20.82077	15.46477	3	36
33	270.6567	249.8417	232.0345	3	352.8
34	712.3336	654.62	662.9693	6.520409	324
35	439.3494	404.6549	392.4677	6.520409	360
36	439.3494	404.6549	392.4677	6.520409	215
37	439.3494	404.6549	392.4677	6.520409	360
38	45.79719	42.62334	33.76551	6.520409	48
39	115.0935	106.6625	91.76123	6.520409	60
40	110.8853	102.7803	88.12707	6.520409	60
41	15.36584	14.37324	10.32592	6.520409	24

42	32.67584	30.45876	23.41075	6.520409	36
43	64.52773	59.96081	48.98032	6.520409	72
44	64.52773	59.96081	48.98032	6.520409	48
45	20.04739	18.72936	13.77956	6.520409	72
46	1640.226	1501.541	1638.561	6.520409	2400
47	2089.383	1910.585	2130.591	6.520409	3240
48	1118.321	1025.576	1081.446	6.520409	2120
49	192.322	177.8117	160.1644	6.520409	370
50	444.5907	409.46	397.5498	10.26826	750
51	944.1499	866.5265	899.9904	6.520409	420
52	180.9169	167.3143	149.8864	10.26826	252
53	68.39128	63.53386	52.16987	6.520409	107
54	2450.814	2239.436	2533.23	10.26826	2300
55	331.7397	305.9395	289.3568	6.520409	400
56	1528.865	1400.05	1518.225	6.520409	973
57	1512.239	1384.895	1500.321	6.520409	1368
58	326.7409	301.3507	284.6296	10.26826	571.4
59	33.02077	30.7788	23.67897	10.26826	98.8
60	61.84511	57.47932	46.77514	10.26826	155

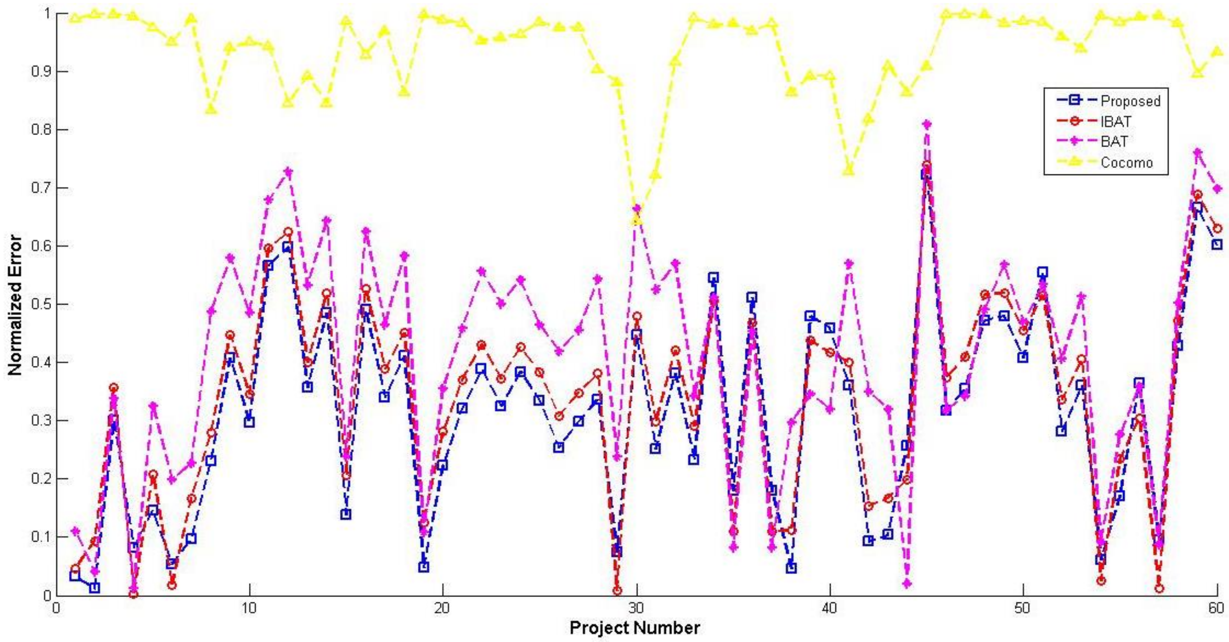

Figure. 2 Analysis of normalized error on Dataset 2

Table 3. Analysis of effort estimation on Dataset 3

Project Number	Hybrid BATGSA Optimization	IBAT Optimization	BAT Optimization	COCOMO	Actual
1	99.20243	93.56455	75.74521	3	117.6
2	93.74146	88.38874	71.37792	3	117.6
3	26.13845	24.48804	18.69947	3	31.2
4	28.0106	26.25112	20.10661	3	36
5	33.69317	31.6061	24.4048	3	25.2

Received: March 26, 2018

6	6.592658	6.133727	4.409609	3	8.4
7	10.98438	10.24602	7.532536	3	10.8
8	280.2345	265.693	225.1002	3	352.8
9	25.39293	23.78613	18.14048	6.520409	72
10	74.65834	70.3147	56.21907	6.520409	72
11	19.86814	18.58797	14.02459	6.520409	24
12	438.1441	416.3439	359.7068	6.520409	360
13	39.85181	37.41485	29.10324	6.520409	36
14	438.1441	416.3439	359.7068	6.520409	215
15	74.65834	70.3147	56.21907	6.520409	48
16	438.1441	416.3439	359.7068	6.520409	360
17	684.2813	651.6947	574.1434	6.520409	324
18	123.0252	116.1592	94.92732	6.520409	60
19	54.41322	51.16594	40.34614	6.520409	48
20	127.3263	120.241	98.41099	6.520409	60
21	73.42792	69.15009	55.2477	3	60
22	280.2345	265.693	225.1002	3	300
23	114.4643	108.0369	88.01115	3	120
24	54.41322	51.16594	40.34614	6.520409	90
25	151.2085	142.9179	117.8548	6.520409	210
26	34.84069	32.68804	25.27728	6.520409	48
27	56.01076	52.67581	41.5894	3	70
28	197.7336	187.1444	156.1494	3	239
29	59.6202	56.08796	44.4047	3	82
30	45.70538	42.94007	33.60213	3	62
31	127.7571	120.6499	98.76027	3	170
32	140.307	132.5641	108.9592	3	192
33	18.05542	16.88392	12.68562	3	18
34	36.37604	34.13594	26.44682	3	50
35	50.43812	47.40997	37.26043	3	60
36	21.69598	20.30703	15.38079	6.520409	42
37	46.49121	43.6821	34.20833	6.520409	60
38	390.2161	370.5845	318.552	6.520409	444
39	27.26034	25.54449	19.54213	6.520409	42
40	58.41479	54.94832	43.46355	6.520409	114
41	825.4542	786.8871	698.9662	3	1248
42	1477.063	1412.18	1286.772	6.520409	2400
43	1370.405	1309.713	1189.493	6.520409	1368
44	1384.298	1323.059	1202.144	6.520409	973
45	338.1075	320.866	274.0882	6.520409	400
46	2139.425	2049.264	1897.824	10.26826	2400
47	887.3909	846.238	754.0714	6.520409	420
48	193.2555	182.885	152.4424	10.26826	252
49	78.77285	74.20987	59.47296	6.520409	107

50	333.4047	316.3806	270.0911	10.26826	571.4
51	40.23975	37.78091	29.40045	10.26826	98.8
52	71.79028	67.60018	53.95606	10.26826	155
53	442.964	420.9472	363.8582	10.26826	750
54	1037.398	990.0667	888.2906	6.520409	2120
55	204.468	193.5508	161.7318	6.520409	370
56	1079.141	1030.109	925.815	3	1181
57	296.004	280.7215	238.4036	3	278
58	2.467429	2.28434	1.573054	3	8.4
59	5388.899	5185.847	5000.907	10.26826	4560
60	1737.146	1662.194	1525.38	6.520409	720
61	296.004	280.7215	238.4036	6.520409	458
62	1311.233	1252.883	1135.682	6.520409	2460
63	390.2161	370.5845	318.552	6.520409	162
64	159.9815	151.2528	125.0365	6.520409	150
65	619.3632	589.5723	517.151	6.520409	636
66	684.2813	651.6947	574.1434	6.520409	882
67	1677.211	1604.562	1470.228	6.520409	444
68	1147.283	1095.493	987.223	6.520409	192
69	654.2464	622.9494	547.741	3	576
70	689.2988	656.4974	578.5597	3	432
71	133.8024	126.3883	103.6672	3	72
72	428.5187	407.1519	351.4232	3	300
73	366.4468	347.901	298.2313	3	300
74	74.65834	70.3147	56.21907	3	240
75	491.4179	467.2367	405.7102	3	600
76	744.7064	709.5443	627.4299	3	756
77	1748.063	1672.694	1535.436	3	1200
78	759.8837	724.0786	640.8482	3	97
79	249.8547	236.7529	199.5761	10.26826	409
80	438.1441	416.3439	359.7068	10.26826	703
81	125.1741	118.1984	96.66707	10.26826	1350
82	217.9967	206.4237	172.9731	6.520409	480
83	164.3846	155.4368	128.6483	6.520409	599
84	91.23059	86.0095	69.37403	6.520409	430
85	759.8837	724.0786	640.8482	10.26826	4178.2
86	272.8409	258.6483	218.8754	10.26826	1772.5
87	296.004	280.7215	238.4036	10.26826	1645.9
88	204.468	193.5508	161.7318	10.26826	1924.5
89	24.46381	22.9115	17.44494	10.26826	648
90	1110.544	1060.239	954.092	10.26826	8211
91	59.6202	56.08796	44.4047	10.26826	480
92	20.59753	19.27386	14.56507	10.26826	12
93	9.271836	8.641216	6.305753	10.26826	38

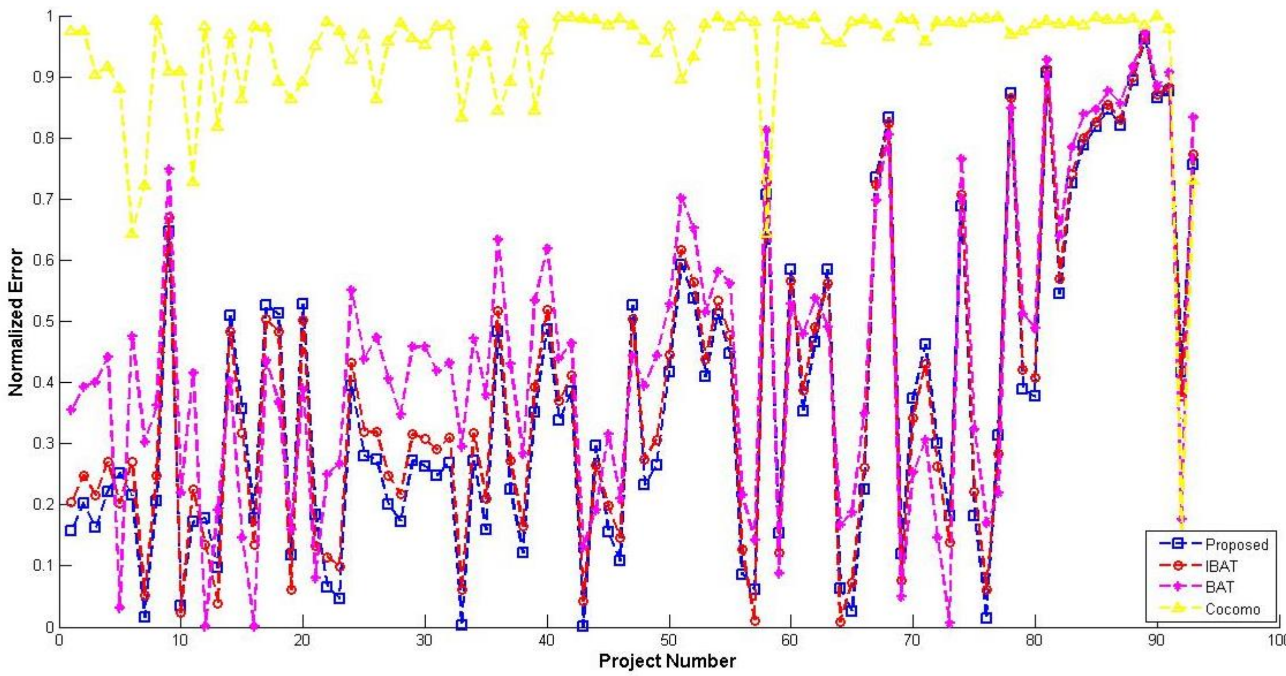

Figure. 3 Analysis of normalized error on Dataset 3

Table 4. Analysis of effort estimation on Dataset 4

Project Number	Hybrid BATGSA Optimization	IBAT Optimization	BAT Optimization	COCOMO	Actual
1	2617.821	4759.931	5605.14	8564.197	287
2	1102.3	1961.087	2138.508	3213.793	82.5
3	6374.61	11853.65	15106.91	17526.39	1107.31
4	2255.341	4085.472	4747.639	5266.707	86.9
5	4273.649	7867.239	9676.503	9519.336	336.3
6	981.854	1741.718	1879.869	2610.139	84
7	203.2856	346.5822	325.2334	520.7564	23.2
8	2583.915	4696.738	5524.325	6818.744	130.3
9	4188.328	7706.258	9461.534	11576.37	116
10	570.9059	998.9934	1027.525	1389.818	72
11	4332.574	7978.464	9825.252	11723.8	258.7
12	1820.726	3280.442	3740.319	5270.448	230.7
13	1768.673	3184.33	3621.389	4542.2	157
14	3664.985	6720.702	8154.226	9598.281	246.9
15	2521.852	4581.12	5376.708	7214.489	69.9

6. Conclusion

This paper implements the hybrid BATGSA algorithm to optimize the performance of the COCOMO model to estimate the software cost. The algorithm has been analyzed on 4 datasets including two datasets of NASA projects downloaded from promise repository. The comparison of effort valueand normalized error on each dataset has been done with the Improved BAT, BAT and COCOMO model. The normalized error has been reduced by 1% on dataset1, 3.5%on dataset2, 0.33% on dataset3 and 46%on dataset4 by BATGSA algorithm as compared to the improved BAT algorithm. The reduction in the normalized error and the convergence of efforts towards the actual effort of

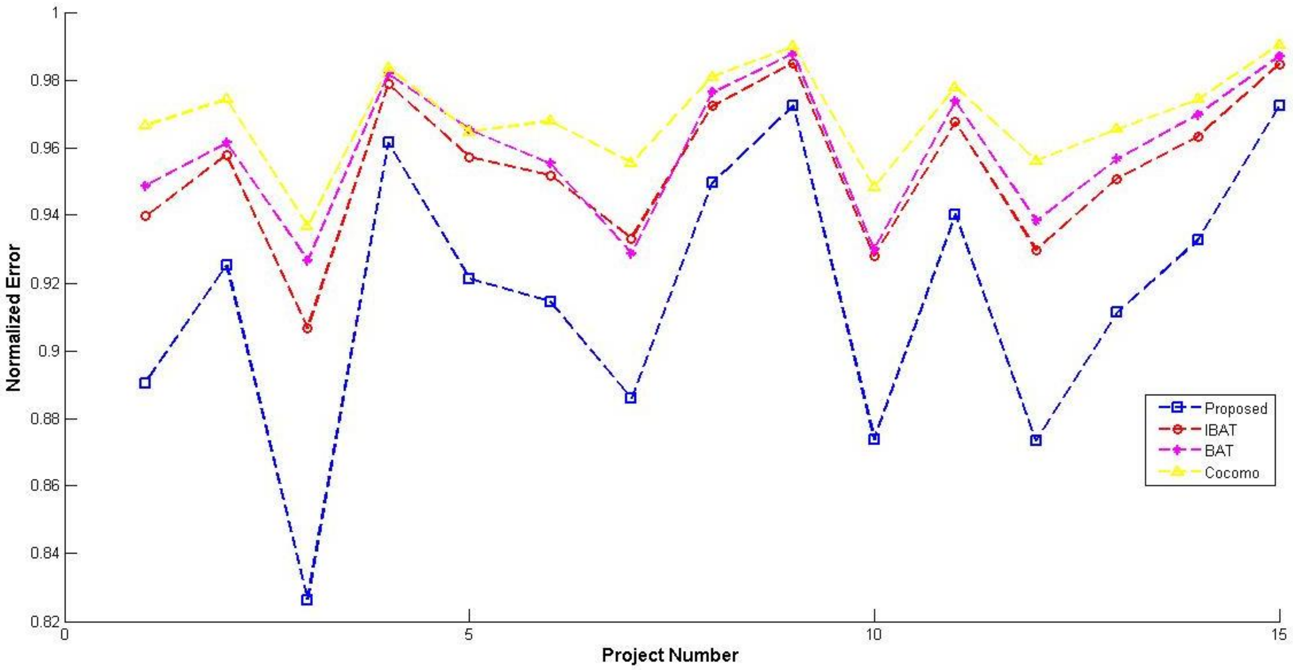

Figure. 4 Analysis of normalized error on Dataset 4

the hybrid BATGSA algorithm as compared to other state of art algorithms proves the significance of the algorithm. In future this algorithm can be applied to optimize different software metrics. Moreover, algorithm can be modified to determine the software quality during the development of software.

References

[1] B. W. Boehm, "Software cost estimation meets software diversity", In: *Proc. of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, pp. 495–496, 2017.

[2] S. Aljahdali and A. F. Sheta, "Software effort estimation by tuning COOCMO model parameters using differential evolution", In: *Proc. of the ACS/IEEE Int. Conf. Comput. Syst. Appl. - AICCSA 2010*, pp. 1–6, 2010.

[3] I. M. Keshta, "Software Cost Estimation Approaches: A Survey", *J. Softw. Eng. Appl.*, Vol. 10, No. 10, pp. 824–842, 2017.

[4] R. Wang, P. Peng, L. Xu, X. Huang, and X. Qiao, "A Novel Algorithm for Software Development Cost Estimation Based on Fuzzy Rough Set", *J. Eng. Sci. Technol. Rev.*, Vol. 9, No. 4, pp. 217–223, 2016.

[5] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic Algorithms in Modeling and Optimization", *Metaheuristic Applications in Structures and Infrastructures*, pp. 1–24, 2013.

[6] A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects", *J. Comput. Sci.*, Vol. 2, No. 2, pp. 118–123, 2006.

[7] N. Gupta, "Optimizing Intermediate COCOMO Model Using BAT Algorithm", In: *Proc. of the 2nd Int. Conf. Comput. Sustain. Glob. Dev.*, p. 1649, 2015.

[8] J. D. Altringham, *Bats: biology and behaviour*. Oxford Univesity Press, 1996.

[9] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm", *Studies in Computational Intelligence*, Vol. 284, pp. 65–74, 2010.

[10] K. Kiełkowicz and D. Grela, "Modified Bat Algorithm for Nonlinear Optimization", *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, Vol. 16, No. 10, pp. 46–50, 2016.

[11] S. Girotra and K. Sharma, "Tuning of Software Cost Drivers using BAT Algorithm", In: *Proc. of the 10th INDIACOM - 2016 3rd international conference on computing for sustainable global development*, pp. 1051–1056, 2016.

[12] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm", *Inf. Sci. (Ny).*, Vol. 179, No. 13, pp. 2232–2248, 2009.