International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-5, January 2020

Internal Adaption based Nature Inspired Algorithm for Application in Software Engineering

Shailendra Pratap Singh, Deepak Kumar Singh

Abstract: ***In this paper, new mutation strategies are proposed to improve the accuracy of the cost estimation by COCOMO's tuning parameters using the Internal adaption based mutation operator for differential evolution algorithm (IABMO Algorithm). The proposed method provides more promising solutions to take the lead evolution and helps DE abstain the circumstance of stability. The proposed algorithm applied software cost estimation and improve the performance of the initial phase for software engineering. This approach is used for precise prediction and reduces the error rate for the initial phase of software development phase projects. The software cost estimation based IABMO algorithm has been capable of a better for effort, MRE, MMRE, and prediction.***

Keywords: Evolutionary Algorithm, Software Engineering, Optimization, COCOMO Model.

I. INTRODUCTION

There are various nature-inspired algorithms used to solve real world optimization problems such as Particle Swarm Optimization (PSO) [14], Genetic Algorithm (GA) [19-20] and many more, but none of them guarantee to have an optimum solution better than DE. After rigorous studies and research, engineers and scientist were able to design a robust algorithm that acquires low cost and better convergence rate.

Internal adaption based mutation operator for differential evolution algorithm (IABMO Algorithm). This is a newly designed DE-variant that introduces solution by merging multiple local optima solutions toward the global optimal solution by enhancing diversity with improved convergence rate. This algorithm applied the software development, due to problem factors in accurate cost estimation. Now a software company is getting more and more worried about the precise prediction of software costs so that good quality software can be produced within time and in the budget. Therefore, proper and stable cost estimation of software development [11-14, 21] in the field of software engineering is a continuous challenge. This type of problem solved by meta-heuristic algorithm that is called the IABMO algorithm.

In this paper, IABMO algorithm is used as a optimization algorithm that is used to tune the parameters of the COCOMO model to get a better effort estimate and minimize the error rate. The performance of the developed model was tested on the NASA Software Project Dataset [22] and compared to the models presented in [13-14].

The rest of this paper is organized as follows, section 2 related work is given, in section 3 proposed approach Internal adaption based operator is explained, in section 4 proposed algorithm applied in software cost estimation model and result analysis have been given, and in section 5 conclusion and future work of this paper is described.

II. RELATED WORK

The differential evolution (DE) is a meta-heuristic algorithm, similar to genetic algorithms, as both methods use the same type of operation such as crossover and mutation. This is a stochastic search trend to guide during the search process to achieve optimized results and it is worth mentioning that it does not use domain derivatives. In this, we can say that it is a population-based and derivative-free method. Algorithm 1 describes the basic DE algorithm [1-10].

Basic Differential Evolution
[1] DEA
initialization of population

$$\vec{\alpha}_{i,G} = \{\vec{\alpha}_{1,i,G}, \vec{\alpha}_{2,i,G}, \ldots\ldots \vec{\alpha}_{D,i,G}\} \quad (1)$$

where, NP is a population size of parameter vectors, i = 1, 2, 3.....NP, D is a dimension and G is a generation.
fitness value of the population
itr = 1
while $FunEvs < MaxFunEvs$
Apply generate a donor vector (mutation strategy):

$$\vec{\gamma}_{i,G} = \vec{\alpha}_{\text{best},G} + \delta_1 \cdot (\vec{\alpha}_{r_1^i,G} - \vec{\alpha}_{r_2^i,G})$$

Where $\vec{\alpha}_{\text{best}}$ denotes the best vector of current population. $\vec{\alpha}_{r_1^i,G}$ and $\vec{\alpha}_{r_2^i,G}$ will be generated from the entire search space
Apply crossover generate the trail vector

$$\vec{\beta}_{j,i,G+1} = \begin{cases} \vec{\gamma}_{j,i,G+1}, & \text{ifr(j)} \le \text{Cror j} = \text{rn(i)} \\ \vec{\alpha}_{j,i,G}, & \text{ifr(j)} > \text{Crand j} \ne \text{rn(i)} \end{cases} \quad (2)$$

Manuscript published on January 30, 2020.
* Correspondence Author
Shailendra Pratap Singh*, Department of Computer Science and Engineering, Bundelkhand Institute of Engineering and Technology, Jhansi, U.P., India. E-mail: shail2007singh@gmail.com
Dr. Deepak Kumar Singh, Director, Sachdeva Institute of Technology, Mathura, UP, India. Email: yadav.k.deepak@gmail.com

where Cr is the crossover rate (constant) ε [0,1], rn(i) ε (1,2,...D) is a randomly chosen index which ensures that $\vec{\beta}_{j,i,G+1}$ gets

at least one element from $\vec{\gamma}_{j,i,G+1}$. j = 1,2,3,.... D, r (j) ε [0,1], j is the uniform random generator number.

Apply the better fitness function according to the selection strategy

itr = itr + 1

end while

III. INTERNAL ADAPTION BASED MUTATION OPERATOR WITH DE ALGORITHM (IABMO)

Differential Evolution sustains from the problem [23] and [24] of stagnation in local optima in the global space. After, large number of iteration we have seen that the generated solution losses its diversity and start stagnating. In DE, the major amenability of diversification maintenance goes to the mutant operator, a proposed method has given attention to this operator strategy.

To solve the issue related to *loss_of_diversity* , a new approach designated as an internal adaption based mutation operator has been proposed. The operator used Internal based adaption, that is to maintain the diversity among solutions. Design the internal vector using multiplying adaption factor with some self generated random value, which actively aggregates improvement in *loss_of_diversity* and effectively improves convergence rate. This algorithm is similar to the environmental event because the DE algorithm keeps the internal environment constant.

3.1 Internal adaption based mutation operator (IABMO)

The internal environment (IE) evolution is a contribution as the approach that maintains the internal environment of the global search. This internal environment have the different parameter values. This parameter has used the balance of the environment and maintain the diversity of the nature of the problem. IE maintains the counterbalancing resources, they constantly interact with them and Adapt situations in the environment).

In the same way, Internal environment (IE) is designed and multiplied with each vector as mutation strategy, Thus, maintaining diversity from early generations to end. This method IE used multiply these vectors to get more area for exploration while others use it for exploitation. These enhanced mutation strategies have been named after their original mutation method, which is a normal procedure for creating a new version of DE, which is called the internal environment or internal vector.

The new vectors are created based on the environment according to the current environment of search space. These vectors are used to maintain the environment adaption condition, Which is called the internal environment (IE) based vectors. IE has explained the equation (3):

$$IE = (current\_vector - random\_vector) * AF \quad (3)$$

where, *current_vector* represents the current population of search space in globally and *random_vector* is represent the given population in the search area. IE retains the diversity of the environment of search space. Due to the requirement of adaptation factor (AF) (0.1 to 1), the required diversity is achieved. This place is selected according to the fitness function and current environment to enhanced the convergence speed in the search field.

In order to generate DE optimum based mutation operator of DE/best/1, internal adaption optimization is used to find the current candidate solution for the search space. We define the general mutation operator in the equation (4), and the internal environment (IE) based mutation operator in the equation (5).

$$\vec{\gamma}_{i,G} = \vec{\alpha}_{\text{best},G} + \delta_1 \cdot (\vec{\alpha}_{r_1^i,G} - \vec{\alpha}_{r_2^i,G}) \quad (4)$$

Generate the new mutant vector:

$$\vec{\gamma}_{i,G} = \vec{\alpha}_{\text{best},G} + \delta_1 \cdot (\vec{IE}_{r_1^i,G} - \vec{IE}_{r_2^i,G}) \quad (5)$$

where, $\vec{\alpha}_{\text{best}}$ denotes the best vector of current population. $\vec{\alpha}_{r_1^i,G}$ and $\vec{\alpha}_{r_2^i,G}$ will be generated from the entire global search space. $\vec{IE}_{r_1^i,G}$ and $\vec{IE}_{r_2^i,G}$ will be generated from the better environment of global search. IEs is environment adaption based vectors. This vectors apply for current circumstance of the mutation process in the DE algorithm. δ_1 is mutation factor.

The new environmental-based mutation strategy maintains a suitable environment according to the candidate's solution. When this environment provides enough variety, then this mutation strategy will be better performance to trap in search space. This mutation strategy will generate better candidate solutions. Therefore, crossover and selection operator is not changed for the proposed algorithm.This process is explained in algorithm 2.

IABMO Algorithm

[1] Set initialization of parameters as given:

δ_1 = rand/4, where rand value (0.1 to 1)

Cr = 0.25

AF = 0.1 to 1 Set the Population Size = 80*D Fitness function of candidate solution itr = 1 while $FunEvss < MaxFunEvs$ Apply Internal adaption based mutation operator and select best donor vector:

$$\vec{\gamma}_{i,G} = \vec{\alpha}_{\text{best},G} + \delta_1 \cdot (\vec{IE}_{r_1^i,G} - \vec{IE}_{r_2^i,G})$$

Apply trail vector using eq (2)

Apply the better fitness function according to the selection strategy itr = itr + 1 end while

Table 1: Control Parameters

Sr. No.	Parameter	Type
1	NP (Population Size)	80
2	Scale factor (Mutation)	δ [0-1.5]

3	CR (Crossover Rate)	[0-1]
4	D (Dimension)	[10, 20 & 40]
5	FEs	Function Evaluations
6	FunEvs	Function Evaluations
7	Adaption Factor (AF)	[0.1-1]

IV. PROPOSED ALGORITHM APPLIED IN SOFTWARE PROJECTS

The proposed meta-heuristic algorithm gives better results for improving accuracy in the software cost estimation model. This algorithm is applied to software cost model (COCOMO). This meta-heuristic approach is applied the cost estimation for embedded projects for minimizing the effort and error in during of process. We can calculate the Effort and duration of time for projects. With these calculations, we can find out how many employees are required to complete a project on time. [15-18] Software cost attempts are shown in the given equation below:

$$Effort(E) = a(KLOC)^b EAF \quad (6)$$

$$DevelopTime(T) = c(E)^d EAF \quad (7)$$

Where, E is the effort in person-months, T is the developing time of software project in months, EAF is the effort adjustment factor, a , b , c and d are constants based on the model using Table 2.

Table 2: Intermediate COCOMO based Parameters

Software project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semi-detached	3	1.12	2.5	0.35
Embedded	2.8	1.2	2.5	0.32

This approach is used to an equation 6 and 7 to assess software cost attempts and scheduling using the COCOMO model. Where E is an effort of project and, which is unit measurement in person-months, and T is time schedule of duration of project, which is unit measurement in months, a, b, c, and d are constants according to project name of the COCOMO model. These constant values are determined using prediction analysis applied to historical data. These constants values applied the proposed algorithm and generate the effort, MRE , MMRE, VAF, prediction, Time, and convergence speed. The value of a, b, c, and d differ for different types of projects(Semidetached).

4.1 IABMO Algorithm based Software Cost Estimation (SCE) Model

The IABMO algorithm applied the cost estimation of different projects. This algorithm improves the performance of semidetached project of the COCOMO Model. The accuracy of this model can be further improved by the proposed algorithm has used in constructive cost Model (COCOMO) for SCE. This algorithm optimizes the parameters of COCOMO model like a, b, c, and d. These parameters value has used the performance of accuracy, effort, and prediction for semidetached. These values are optimization according to project size (KLOC (Kilo line of code) or LOC (Line of Code).

The algorithm 3 is applied to the DATASET 63, and apply them equation are 6, 7, 8, 9, 10, 11, 12 and 13. These equations is used to the effort, MRE, MMRE, MSE and prediction. According to given equations find the new value (optimize value). These values are optimized according to has applied the effort, MRE, MMRE, MSE and prediction. This process explained in algorithm 3.

COCOMO Model Based on IABMO Algorithm

[1] Population based initial values of parameters as

δ_1 = rand/3, where rand value (0 to 1)

Cr = 0.4;

AF = 0.1 to 1; Initial set the 63 projects (dataset). This dataset is given the kilo line of code (KLOC) according to the project size (LOC (Line of Code))

$$Effort(E) = a(KLOC)^b EAF \qquad Time(T) = c(E)^d$$

$$MRE = |AE - EE| / AE$$

$$VAF = [1 - var(AE - EE)/var(AE)]$$

$$MMRE = 1/N \sum_{i=1}^{N} |AE - EE| / AE$$

$$MSE = 1/N \sum_{i=1}^{N} (AE\_EE)^2$$

$$Prediction(N) = MMRE \leq 25/100$$

$$Convergence = \sqrt{1/N \sum (AE - EE)}$$ PopSize = 50*D

Initialize of fitness function of population (randomly) itr = 1 while $FEs < MaxFEs$ Apply Internal adaption based mutation operator and select best donor vector:

$$\vec{\gamma}_{i,G} = \vec{\alpha}_{best,G} + \delta_1 \cdot (\vec{IE}_{r_1^i,G} - \vec{IE}_{r_2^i,G})$$

Apply crossover operator Apply Selection operator itr = itr + 1 end while

4.2 Testing Framework

In this paper, the proposed algorithm incorporates the COCOMO model and tuning the parameter of optimization based software model. This algorithm is applied to calculate the parameters like a, b, c, and d of COCOMO model, given Table 2. Estimated parameters will make the calculation of efforts for all projects (Semidetached) significantly important. This estimate the parameters of the COCOMO model presented in the IABMO Algorithm. This algorithm is used in equations 6, 7, 8, 9, 10, 11, 12, and 13. These equations calculate the effort, Time, MRE, MMRE, VAF, Prediction, and convergence speed of projects. These performance criteria of projects, and optimize the minimum error and high prediction, this process is explained Algorithm 3. These parameters are given in Table 3 to manage the development process of software projects. This COCOMO based IABMO algorithm is tuning parameter and used demonstration on NASA Software Project Dataset [22].

The experiments have been conducted on a PC with Intel Core i7 CPU with a speed of 3.40 GHz, installed memory (RAM) 8 GB, operating system Windows 10 Pro 64 bit & x64 based processor.

4.2.1 Evaluation Criteria

IABMO algorithm checked the benchmark function on COCO Platform. This algorithm is not check the real world application. It apply the software cost estimation model. To verify the performance of the software developed model [21]. There are six performance criteria given below:

1. Magnitude of Relative Error(MRE): MRE is measurement of error in the software project defined in [13] and [19]:

$$MRE = | AE - EE | / AE \quad (8)$$

where AE is denoted Actual Effort and EE is represented Estimated Effort.

2. Variance-Accounted-For(VAF): VAF is measurement of close value for actual effort and estimated effort of projects.This value used the correctness of projects or models. It is defined as [13]:

$$VAF = [1 - var(AE - EE)/var(AE)] \quad (9)$$

where AE is denoted Actual Effort and EE is represented Estimated Effort.

3. Mean Magnitude of Relative Error(MMRE): The MMRE [13, 19, 21] is the measurement of the actual error of the software project. This error has used the cost and developing time of the project. The MMRE error is minimum for the project, it means to minimize the effort and time of developing projects. MMRE is explained in equation 10.

$$MMRE = 1/N \sum_{i=1}^{N} | AE - EE | / AE \quad (10)$$

where AE is denoted Actual Effort and EE is represented Estimated Effort.

4. MSE(Mean Squared Error): MSE is the mean squared error of the projects. This type of error finds the accuracy of the project for developing during the initial phase. MSE is a minimum error that can increase the accuracy of projects. It can be calculated [11] by the following equation 11 as described below:

$$MSE = 1/N \sum_{i=1}^{N} (AE - EE)^2 \quad (11)$$

where AE is denoted Actual Effort and EE is represented Estimated Effort.

5. Prediction(N): Predictions [13, 19] are a measurement of the accuracy of the project. This is used the actual project of 25% of the total value project. Further, this value is higher which means the project is accurate. Prediction is explained in equation 12.

$$Prediction(N) = MMRE \leq 25/100 \quad (12)$$

6. Convergence(C): Convergence [21] defined as speed of algorithm for the parameter. These parameter checked that how fast an algorithm approaches to the desired value. Convergence process can be calculated using the actual effort and estimated effort as shown in the equation below:

$$C = \sqrt{1/N \sum (AE - EE)} \quad (13)$$

Table 3: Parameters of IABMO Algorithm based COCOMO

Sr. No.	Parameter Name	Description	Value
1	itr	No. of maximum iteration	100
2	F	Mutation Factor	0.1 to 2.0
3	Cr	Crossover Rate	0.1 to 1.0
4	D	Dimension	3
5	Pop	Population Size	100
6	a	Parameter for a	-5 to +5
7	b	Parameter for b	-5 to +5
8	c	Parameter for c	-5 to +5
9	d	Parameter for d	-15 to +15

Table 4: Comparison between effort, MRE, and schedule of time for semidetached projects

Sr No	LOC	Actual_Et	KLOC	IABMO_E	GA_E	PSO_E	DE_E	IABMO_MRE	GA_MRE	PSO_MRE	DE_MRE	IABMO_T	GA_T	PSO_T	DE_T
1	113	2040	0.113	0.226203738	0.273968348	0.267897577	0.259392461	0.999889116	0.999865702	0.999868678	0.999872847	1.426547259	1.59730417	1.58293396	1.559548065
2	293	1600	0.293	0.689656945	0.792034851	0.781005391	0.757652698	0.999546595	0.999504978	0.999511872	0.999526467	2.107331367	2.316076983	2.301985759	2.269489891
3	132	243	0.132	0.271312309	0.325764581	0.318982031	0.30895112	0.998728062	0.998659405	0.998687317	0.998728596	1.520286944	1.697105499	1.68264267	1.657964062
4	60	240	0.06	0.107853678	0.135324525	0.131590806	0.127251895	0.999446052	0.999436148	0.999451705	0.999469784	1.100787471	1.247884148	1.234248657	1.215475423
5	16	33	0.016	0.022973035	0.03103065	0.029825548	0.028765974	0.999020617	0.999059677	0.999096196	0.999128304	0.640674847	0.745279704	0.734141329	0.722306246
6	4	43	0.004	0.004537412	0.006621793	0.006287458	0.006047301	0.999829473	0.999846005	0.99985378	0.999859365	0.363156739	0.434044202	0.425735626	0.418466068
7	6.9	8	0.0069	0.008587197	0.01215643	0.011598163	0.011167329	0.998357382	0.998480446	0.99855023	0.998604084	0.454003545	0.53687676	0.5274848	0.518675765
8	22	1,075	0.022	0.033345139	0.044247398	0.042648363	0.041159455	0.999957729	0.99995884	0.999960327	0.999961712	0.729915572	0.843826573	0.832031079	0.818800422
9	30	423	0.03	0.047932476	0.062512791	0.06041835	0.058345251	0.999850295	0.999852216	0.999857167	0.999862068	0.828765559	0.952317139	0.939902541	0.925157388
10	29	321	0.029	0.046068456	0.060195529	0.058161372	0.056161907	0.999809753	0.999812475	0.999818812	0.999825041	0.817339563	0.939809806	0.927461408	0.912889767
11	32	218	0.032	0.051692016	0.06717358	0.064959866	0.062739034	0.999688748	0.999691864	0.999702019	0.999712206	0.850960609	0.9765893	0.964049899	0.948968793
12	37	201	0.037	0.061262401	0.078967929	0.076463158	0.0738705	0.99960569	0.999607125	0.999619586	0.999632485	0.903086077	1.033475656	1.020662132	1.004797523
13	25	79	0.025	0.038724677	0.051020551	0.049232096	0.047525491	0.999340482	0.99935417	0.999376809	0.999398412	0.769142792	0.886958479	0.874904469	0.861069102
14	3	60	0.003	0.003240633	0.004805835	0.00455165	0.004375277	0.999910169	0.999919903	0.999924139	0.999927079	0.322798639	0.387982021	0.380218132	0.373650546
15	3.9	61	0.0039	0.004404976	0.006437609	0.006111211	0.005877488	0.999883005	0.999894465	0.999899816	0.999903648	0.35941111	0.429779889	0.42152007	0.414315152
16	6.1	40	0.0061	0.007434194	0.010596804	0.010099203	0.009721654	0.999712061	0.99973508	0.99974752	0.999756959	0.431661294	0.511686195	0.502544132	0.494108983
17	3.6	9	0.0036	0.004011178	0.005888337	0.005585852	0.005371362	0.999272121	0.99934574	0.99937935	0.999403182	0.347821493	0.416571842	0.408465153	0.401460884
18	320	11,400	0.32	0.764580949	0.873772747	0.862273659	0.836638463	0.99993007	0.999923353	0.999924362	0.999926611	2.184788781	2.397076873	2.383139765	2.349643268
19	1,150	6,600	1.15	3.415186862	3.634041836	3.62681471	3.528004833	0.999525255	0.999449388	0.999450483	0.999465454	3.68898005	3.947562017	3.940103231	3.888202738
20	299	6400	0.299	0.706209066	0.810127225	0.798988334	0.775129361	0.999884163	0.999873418	0.999875158	0.999878886	2.124897045	2.334458286	2.320400097	2.287676745
21	252	2455	0.252	0.578144809	0.669577531	0.659378073	0.639469315	0.999748519	0.99972726	0.999731414	0.999739524	1.981180581	2.183849942	2.169555662	2.138703548
22	118	724	0.118	0.237957801	0.287508919	0.281245245	0.272339954	0.999621354	0.999602888	0.99961154	0.99962384	1.452065553	1.624502751	1.610102548	1.586363349
23	77	539	0.077	0.144408297	0.178685106	0.174136888	0.168479152	0.999677882	0.999668488	0.999676926	0.999687423	1.219181862	1.375382242	1.361399209	1.340926119
24	90	453	0.09	0.17332521	0.212606919	0.207479901	0.200801487	0.999547103	0.999530669	0.999541987	0.99955673	1.299609414	1.461653989	1.447489202	1.425877169
25	38	523	0.038	0.063204035	0.081349572	0.078787745	0.076120327	0.999844072	0.999844456	0.999849354	0.999854454	0.91300241	1.044279696	1.031416933	1.015404113
26	48	387	0.048	0.083071154	0.105535695	0.102422555	0.099001208	0.999729433	0.999727298	0.999735342	0.999744183	1.004659344	1.14388475	1.130608335	1.113237595
27	9.4	88	0.0094	0.012329844	0.017156135	0.01641286	0.015812951	0.999792115	0.999805044	0.99981349	0.999820307	0.515283691	0.605674506	0.595645983	0.585825429
28	13	98	0.013	0.018018213	0.024621589	0.023622186	0.022773531	0.99973591	0.999748759	0.999758957	0.999767617	0.588451052	0.687310714	0.676605851	0.665601546
29	2.14	7.3	0.00214	0.00218264	0.003298431	0.003114701	0.002991986	0.999485627	0.99954816	0.999573329	0.999590139	0.281096204	0.340094872	0.332942662	0.327114315
30	1.98	5.9	0.00198	0.001992949	0.003024856	0.002854412	0.002741526	0.99941435	0.999487312	0.999516201	0.999535335	0.272292034	0.329943175	0.322927156	0.317256878
31	62	1063	0.062	0.112071781	0.140359951	0.13652669	0.132033687	0.999870466	0.999867959	0.999871565	0.999875791	1.1156679	1.263943358	1.25025865	1.231270161
32	390	702	0.39	0.963703837	1.08924249	1.076780662	1.045181668	0.998596669	0.998448373	0.998466124	0.998511137	2.369144213	2.589322168	2.575836294	2.539983036
33	42	605	0.042	0.071055822	0.09094624	0.08815981	0.085192145	0.99984997	0.999849676	0.999854281	0.999859187	0.951198431	1.085843286	1.072799217	1.056217929
34	23	230	0.023	0.035125261	0.046494068	0.044831374	0.043270102	0.999792805	0.999797852	0.999805081	0.999811869	0.743323888	0.858581703	0.846695868	0.833257943
35	13	82	0.013	0.018018213	0.024621589	0.023622186	0.022773531	0.999684381	0.999699737	0.999711925	0.999722274	0.588451052	0.687310714	0.676605851	0.665601546
36	15	55	0.015	0.021302215	0.028877612	0.027740365	0.026751416	0.99945158	0.999474953	0.99949563	0.999513611	0.623964543	0.726756514	0.715752681	0.704182228
37	60	47	0.06	0.107853678	0.135324525	0.131590806	0.127251895	0.997171332	0.997120755	0.997200196	0.997292513	1.100787471	1.247884148	1.234248657	1.215475423
38	15	12	0.015	0.021302215	0.028877612	0.027740365	0.026751416	0.99748641	0.997593532	0.997688303	0.997770715	0.623964543	0.726756514	0.715752681	0.704182228
39	6.2	8	0.0062	0.007576982	0.010790541	0.010285314	0.00990113	0.998535035	0.998651182	0.998714336	0.998762359	0.434545181	0.514941166	0.505766276	0.497282705
40	3	8	0.003	0.003240633	0.004805835	0.00455165	0.004375277	0.999326267	0.999399271	0.999431044	0.99945309	0.322798639	0.387982021	0.380218132	0.373650546
41	5.3	6	0.0053	0.00630668	0.009060425	0.008624293	0.008299548	0.998348487	0.998489929	0.998562618	0.998616742	0.407513039	0.484389238	0.475528932	0.467501222
42	45.5	45	0.0455	0.078031747	0.099429828	0.09645139	0.093219533	0.997802556	0.997790448	0.997856636	0.997928455	0.982892923	1.120271607	1.107086854	1.090036686
43	28.6	83	0.0286	0.045325881	0.059271159	0.05726124	0.055291184	0.999275077	0.99928589	0.999310106	0.999333841	0.812704064	0.93473324	0.922412084	0.907910947

44	30.6	87	0.0306	0.049055993	0.063907408	0.061777006	0.059659651	0.999256535	0.999265432	0.999289919	0.999314257	0.835513488	0.959699811	0.947246744	0.9323993
45	35	106	0.035	0.057406043	0.074226845	0.071837318	0.069393797	0.999295461	0.999299747	0.999322289	0.999345342	0.882767641	1.011320568	0.998610801	0.983050706
46	73	126	0.073	0.1356706	0.16837388	0.164011109	0.158665432	0.998698504	0.998663699	0.998698325	0.998740751	1.192837413	1.347065274	1.333151067	1.313053747
47	23	36	0.023	0.035125261	0.046494068	0.044831374	0.043270102	0.998676255	0.998708498	0.998754684	0.998798053	0.743323888	0.858581703	0.846695868	0.833257943
48	464	1272	0.464	1.180929673	1.321888191	1.308764137	1.270798897	0.999067293	0.99896078	0.998971097	0.999000944	2.543839192	2.770834852	2.757878992	2.719822623
49	91	156	0.091	0.175580559	0.215240657	0.210070552	0.203313243	0.998669216	0.998620252	0.998653394	0.99869671	1.305503352	1.467965999	1.453789519	1.432094495
50	24	176	0.024	0.036918594	0.048751923	0.047026095	0.045392254	0.999716619	0.999723	0.999732806	0.999742089	0.756392205	0.872950469	0.860978572	0.84733921
51	10	122	0.01	0.013255557	0.01838063	0.017593883	0.016952904	0.999839787	0.999849339	0.999855788	0.999861042	0.528506753	0.620466909	0.610309759	0.600273439
52	8.2	41	0.0082	0.010508971	0.014734377	0.014079092	0.01356078	0.999614473	0.999640625	0.999656608	0.999669249	0.487256132	0.574260061	0.564514172	0.555153819
53	5.3	14	0.0053	0.00630668	0.009060425	0.008624293	0.008299548	0.999292209	0.999352827	0.999383979	0.999407175	0.407513039	0.484389238	0.475528932	0.467501222
54	4.4	20	0.0044	0.005072682	0.007363688	0.00699776	0.006731756	0.999594003	0.999631816	0.999650112	0.999663412	0.377610804	0.450480364	0.441986891	0.434468824
55	6.3	18	0.0063	0.007720162	0.010984635	0.010471795	0.010080968	0.999337662	0.999389743	0.999418234	0.999439946	0.43740173	0.518164265	0.50895703	0.500425542
56	27	958	0.027	0.042373428	0.05558862	0.053676378	0.05182369	0.999940946	0.999941974	0.99994397	0.999945904	0.79376876	0.913981809	0.901774275	0.887561814
57	17	237	0.017	0.024661713	0.03319912	0.031926832	0.030796342	0.999854491	0.999859919	0.999865288	0.999870058	0.656779176	0.763109378	0.751844908	0.739755819
58	25	130	0.025	0.038724677	0.051020551	0.049232096	0.047525491	0.999599216	0.999607534	0.999621292	0.999634419	0.769142792	0.886958479	0.874904469	0.861069102
59	23	70	0.023	0.035125261	0.046494068	0.044831374	0.043270102	0.999319217	0.999335799	0.999359552	0.999381856	0.743323888	0.858581703	0.846695868	0.833257943
60	6.7	57	0.0067	0.008296702	0.011764485	0.011221314	0.010803843	0.9997766	0.999793606	0.999803135	0.999810459	0.44856788	0.530753682	0.521421597	0.512703262
61	28	50	0.028	0.044215332	0.057887377	0.055913949	0.053987956	0.998823621	0.998842252	0.998881721	0.998920241	0.805678464	0.927036518	0.914757078	0.900362922
62	9.1	38	0.0091	0.011870703	0.016547192	0.015825782	0.015246342	0.999535005	0.999564548	0.999583532	0.99959878	0.508484819	0.598061708	0.58810048	0.578391198
63	10	15	0.01	0.013255557	0.01838063	0.017593883	0.016952904	0.998696933	0.998774625	0.998827074	0.998869806	0.528506753	0.620466909	0.610309759	0.600273439

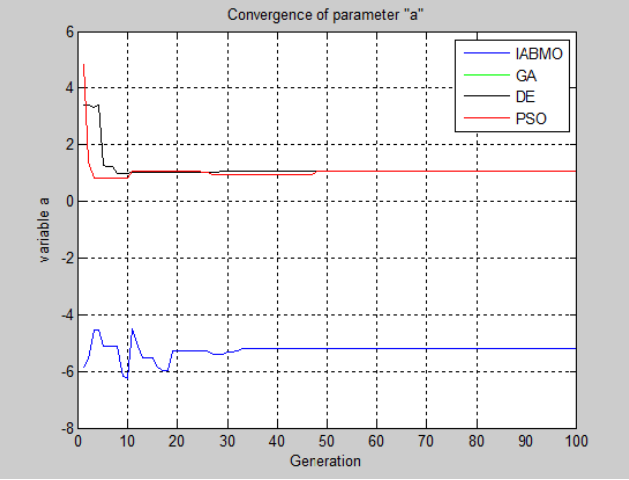

Figure 1: Comparison of variable a

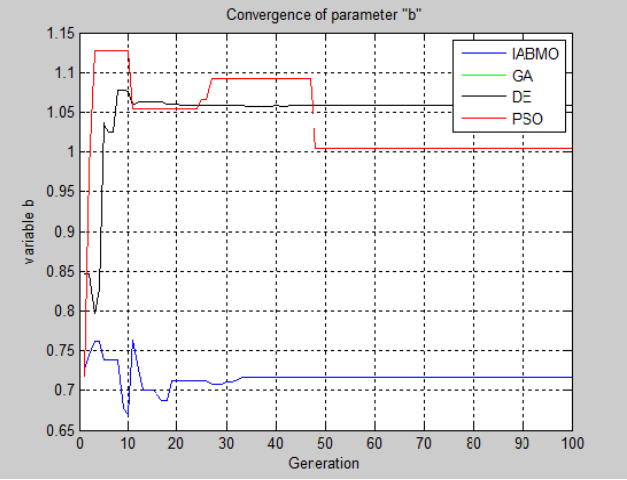

Figure 2: Comparison of convergence b

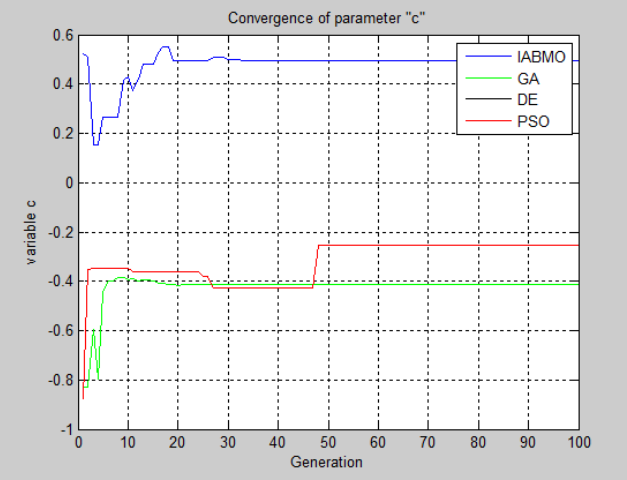

Figure 3: Comparison of variable c

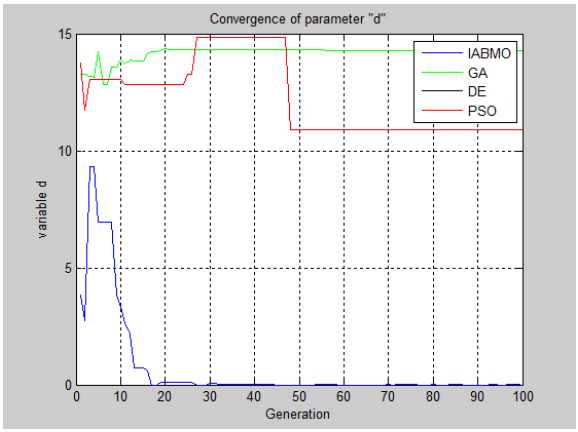

Figure 4: Comparison of convergence d

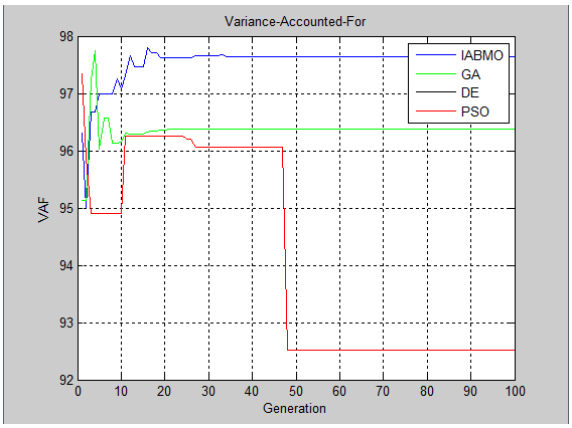

Figure 5: Comparison of VAF

Figure 6: Comparison of MMRE, MSE, and Prediction for Semidetached projects

2* Sr No	2*Performance Evolutions	Algorithms			
		IABMO	GA	PSO	DE
1	MMRE	0.8015	0.9647	0.9326	0.9278
2	MSE	2.3019	2.2575	2.4901	2.3918
3	Pridiction	0.5420	0.4912	0.4976	0.5307

4.3 Experimental Results and Analysis

Used on the data set presented by the NASA project 63 [22], to find out the strength of the developed IABMO algorithm based COCOMO model. The newly created IABMO algorithm is compared with the existing COCOMO algorithms such as GA [13], PSO [14], and DE [11], which creates a suitable model structure for using better estimates of software effort for NASA Software Projects 63 is:

1. Result analysis for Table 4 and 5: In this table represent the data set of NASA project 63. Table 4 represent the columns like LOC stand for line of code, Actual_E stand for actual effort for given data set, IABMO_E stand for internal adaption optimization based mutation operator for differential evolution algorithm with effort, GA_E stand for Genetic algorithm based effort, PSO_E stand for Particle swarm optimization based effort, DE_E stand for differential evolution effort.

IABMO_MRE stand for internal adaption optimization based mutation operator for differential evolution algorithm with mean relative error, GA_MRE stand for Genetic algorithm based mean relative error, PSO_MRE stand for Particle swarm optimization based mean relative error, DE_MRE stand for differential evolution based mean relative error.

IABMO_T stand for internal adaption optimization based mutation operator for differential evolution algorithm with developing time, GA_T stand for Genetic algorithm based developing time, PSO_T stand for Particle swarm optimization based developing time, DE_T stand for differential evolution based developing time.

2. Proposed approach comparison between evolutionary algorithms of MRE for semidetached Projects: The error calculated from the algorithm 3 is done with the actual data set of the NASA Project 63 and we have calculated the MRE using Equation 8. The result of the comparison is given as the value of Table 4. Proposed algorithm MRE is better, COCOMO based algorithms like PSO, GA, and DE. This proposed approach is minimize the error rate of projects like semidetached.

3. Variance-Accounted-For(VAF): To verify the result of proposed approach is better then COCOMO based algorithms like PSO, GA, and DE. In figure 5 shows the VAF performance according to IABMO algorithm. This algorithm is used to cocomo parameter. This parameter show the generation as represent X-axis and achieved the target value VAF (98%) in Y axis. VAF is represent the close value of actual effort and estimated effort.

4. Mean Magnitude of Relative Error(MMRE) and Prediction: Another widely used error counting method, called MMRE, is shown in the 10 equations under consideration in the project, the result of which has been displayed in Table 5, which clearly shows minimize the MMRE than other cost estimation techniques. Shows, having a better MMRE, better prediction results are shown in Table 5 which is generated according to equation 12.

5. Comparison MSE for COCOMO based algorithms: We calculated the MSE using equation 11. This equation find the accuracy of the projects between the actual effort and estimation effort. The mean squared error (MSE) is find the squared error for COCOMO model. This error minimization means that is better accuracy of project. Table 5 which clearly shows a decreased MSE as compared with COCOMO based algorithms.

6. Convergence(C): The rate of convergence which shows that according to calculation from equation 13, the estimated value reaches to the desired value, this figure is also plotted in 1, 2, 3, and 4, which show that the proposed IABMO has better convergence rate than the other evolutionary algorithms like GA, PSO and DE.

In figure 1 shows the convergence parameter 'a', its measurement of effort (duration of developing effort for projects) for constants value 'a' of variable projects. This variable measurement of performance according to projects sizes (KLOC).

The convergence value 'a' is minimum, which better performance for convergence speed.

This convergence speed of the algorithm is represented as a generation (1 to 100) is shown in figure 1. In figure, X axis is denoted the 'generation' and Y axis is denoted the 'variable a'. Similar shows the convergence speed and generation in the figures 2, 3, and 4.

We can see that the IABMO algorithm has a better performance increase than GA, PSO, and DE. Due to the low generation of results and diversity of our experiment, the complexity of a project increases in cost estimation of project, but after the internal adaption optimization based mutation operator generated by the tuning parameters of the software cost estimation model. This operator has been capable of a better effort, MRE, MMRE, MSE, and prediction. This operator also will be helpful in reducing the error rate. We can see variation in diversity, which will improve overall results as shown in Tables 4, and 5.

V. CONCLUSION

While applying in this paper, the internal adaption based operator helped maintain the diversity improve the convergence speed on high dimensions in the global (search) space. In the DE algorithm, it begins with a high generation of function evaluation and applies an internal adaption optimization method. The method enhanced the diversity in local and global (search) area. The proposed algorithm applied COCOMO Model (software cost estimation) and improve the performance of the Effort, MRE, MMRE, and prediction. The developed COCOMO based IABMO algorithm has been capable of a better effort, MRE, MMRE, and prediction. Software cost estimation using IABMO algorithm have been seen to have improved performance in all projects(semidetached). This algorithm of the future scope controls the tuning of control parameters by the internal and external dynamic selection of the entire population.

REFERENCES

1. Storn, Rainer, and Kenneth Price, "Differential Evolution -A Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces", International Computer Science Institute, Berkeley. Berkeley, CA (1995).
2. G. H. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, H. K. Chen, "Differential Evolution with Multi-Population Based Ensemble of Mutation Strategies", Information Sciences, DOI: 10.1016/j.ins.2015.09.009, pp. 329-345, 2016.
3. Yong Wang, Zhi-Zhong Liu, Jianbin Li, Han-Xiong Li and Gery G. Yen, Utilizing cumulative population distribution information in differential evolution, Applied Soft Computing, vol. 48, pp. 329-346, 2016.
4. Y. Wang, B. Xu, G. Sun, and S. Yang. A two-phase differential evolution for uniform designs in constrained experimental domains. IEEE Transactions on Evolutionary Computation, in press, DOI: 10.1109/TEVC.2017.2669098.
5. Zhi-Zhong Liu, Yong Wang Shengxiang Yang and Zixing Cai, Differential evolution with a two-stage optimization mechanism for numerical optimization, 2016 IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, pp. 3170-3177, 2016.
6. Shailendra Pratap Singh, and Anoj Kumar, Homeostasis Mutation Based Differential Evolution Algorithm, Journal of Intelligent& Fuzzy Systems, vol. 32, no. 5, pp. 3525-3537, 2017.
7. Zhongbo Hu, Qinghua Su, Xianshan Yang and Zenggang Xiong, "Not guaranteeing convergence of differential evolution on a class of multimodal functions", Applied Soft Computing (41), pp. 479-487, 2016.
8. Yong Wang, Zhi-Zhong LIU, Jianbin LI, Han-Xiong LI, and Jiahai WANG, "On the selection of solutions for mutation in differential evolution", Frontiers of Computer Science, 2016.
9. Yong Wang, Zhi-Zhong Liu, Jianbin Li, Han-Xiong Li and Gery G. Yen, "Utilizing cumulative population distribution information in differential evolution", Applied Soft Computing, vol. 48, pp. 329-346, 2016.
10. Yong Wang, Han-Xiong LI, Tingwen Huang and Long Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting", Applied Soft Computing, vol. 18, pp. 232-247, 2014.
11. Maghsoud Amiri and Javad Pashaei Barbin, New Approach For Solving Software Project Scheduling Problem Using Differential Evolution Algorithm, International Journal in Foundations of Computer Science & Technology (IJFCST), Vol.5, No.1, January 2015
12. Sultan Aljahdali and Alaa F Sheta, Software effort estimation by tuning COOCMO model parameters using differential evolution, ACS/IEEE International Conference on Computer Systems and Applications-AICCSA 2010, pp. 1-6, 2010.
13. Alaa F. Sheta and Sultan Aljahdali, Software Effort Estimation Inspired by COCOMO and FP Models: A Fuzzy Logic Approach, International Journal of Advanced Computer Science and Applications, Vol. 4, No. 11, 2013.
14. CH. V. M. K. Hari and P.V.G.D. Prasad Reddy, A Fine Parameter Tunning for COCOMO 81 Software Effort Estimation using Particle Swarm Optimization, Journal of Software Engineering 5(1) , pp. 38-48, 2011.
15. B. Boehm, Software Engineering Economics, Englewood Cliffs, NJ, Prentice-Hall, 1981.
16. B. Boehm, Cost Models for Future Software Life Cycle Process: COCOMO2. Annals of Software Engineering, 1995.
17. B. Boehm and et all, Software Cost Estimation with COCOMO II. Prentice Hall PTR, 2000.
18. O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, COCOMO based effort estimation for iterative and incremental software development, Software Quality Journal, vol. 11, pp. 265-281, 2003.
19. SP Singh, A Kumar, Multiobjective differential evolution using homeostasis based mutation for application in software cost estimation, pp. 628-650, Applied Intelligence 48 (3), 2018.
20. SP Singh, A Kumar, Software cost estimation using homeostasis mutation based differential evolution, pp. 173-181, Intelligent Systems and Control (ISCO), 2017 11th International Conference, 2017.
21. SP Singh, VP Singh, AK Mehta, Differential Evolution Using Homeostasis Adaption Based Mutation Operator and its Application for Software Cost Estimation, Journal of King Saud University-Computer and Information Sciences, 2018.
22. Singh, Shailendra. (2018). New adaption based mutation operator on differential evolution algorithm. Intelligent Decision Technologies. 12. 1-9. 10.3233/IDT-180343.
23. Singh SP, Kumar A, Differential Evolution Algorithm Using Population Based Homeostasis Difference Vector, pp. 1579-587, Advances in Intelligent Systems and Computing book series (AISC, volume 553), 2017.