Rev. Roum. Sci. Techn.– Électrotechn. et Énerg.
Vol. **69**, *4*, pp. 431–436, Bucarest, 2024

SOFTWARE COST EFFORT AND TIME ESTIMATION USING DRAGONFLY WHALE LION OPTIMIZED DEEP NEURAL NETWORK

SUMITHRA ALAGAR SAMY[1], VIJAYALAKSHMI NAGARAJAN[2], AHILAN APPATHURAI[3], SALINDA EVELINE SUNIRAM[4]

Keywords: Fuzzy inference system; Dragonfly whale lion optimization; NASA 93; Construction cost modeling (COCOMO) II; Constructive rapid application development model (CORADMO).

Effective software development depends on the exact estimation of effort, time, cost, and customer satisfaction. Software project management requires an accurate evaluation of software development's effort, time, and cost, often underestimated or overestimated. So far, methodology has yet to accurately and reliably estimate the cost of software development. To overcome this issue, this paper proposed a constructive rapid application development model based on software cost effort and time estimation approach (CORADMO-based CETA) for accurate software cost estimation. The data requirements, cost drivers, constraints, and priorities are given as input to the fuzzy inference system (FIS). The processed output, such as effort, time, and cost for the nominal plan, shortest schedule plan, and least cost plan, is computed in the FIS. To reduce the effort, time, and cost, the output is optimized by dragonfly whale lion optimization (DWLO), which provides the best-estimated effort, time, and cost as an output for software development. The proposed CORADMO-based CETA model is tested in the NASA 93 dataset using MATLAB. The performance of the CORADMO-based CETA method is measured in terms of Pred (25%), magnitude of relative error, and mean magnitude of relative error, attaining the values of 80.72%, 87.94%, and 98.13%, respectively. Finally, the CORADMO-based CETA model justifies the suitability of dragonfly whale lion optimization with the proposed fuzzy logic.

1. INTRODUCTION

Software projects developed using logical and analytical task-based methods must complete several tasks, including requirements gathering, testing, and maintenance, all within a predetermined budget and schedule. A software project's capacity to be developed successfully depends on accurate effort estimation, which lowers risk and failure and promotes the more efficient development of software systems [1,2]. The main methods of estimating that rely on regression analysis and mathematical derivations include use case point, SLIM, function point, and COCOMO [3–5].

Construction cost modeling (COCOMO) is one effective method for measuring the efforts made in the early stages of a project [6]. When COCOMO assigns initial values to the parameters, it considers the project assets. Furthermore, the COCOMO measure derives from population-based research and incorporates meta-heuristic techniques, effectively assessing the number of tasks associated with software projects [7–9].

Traditional statistical techniques and parametric models frequently show conflicting results when modeling the relationship between project features and development efforts. Given the notable advancements in software effort estimation, methodologies that will remain with the latest tools and techniques while accounting for abrupt changes in software tasks are needed [10–13].

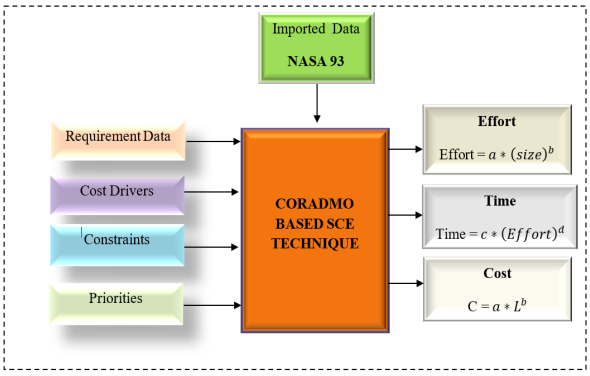


Fig. 1 – Process for software cost estimation.

Software for estimating costs and efforts combines procedures to determine a project's cost and effort in terms of staffing hours and overall time required to complete the task [14]. Precise estimation of time, expense, and effort is critical to project success and enhances the effectiveness of business decision-making [15]. To solve the issues, the CORADMO-based software cost estimation technique proposed in Fig. 1 shows the effort estimation process using CORADMO-based CETA [26,27].

The following primary goals are intended to be attained by the proposed framework,

- The primary purpose of this research is to present a constructive rapid application development model-based software cost effort and time estimation approach (CORADMO-based CETA) for accurate estimation.
- Here, input data about software desires, financial considerations, limitations, and priorities are transmitted to the FIS.
- The processed output from FIS, such as nominal plan, shortest schedule plan, and least cost plan, are optimized using dragonfly whale lion optimization, which provides the proper estimated time, effort, and cost as an output for software development.
- The CORADMO-based CETA model's performance is assessed by Pred (25%), MRE, and MMRE.

The rest of the work is arranged as follows: section 2 explains the related work. Section 3 describes the proposed CETA strategy based on CORADMO. Section 4 provides the findings and a discussion of the proposed approach. Section 5 contains closing remarks.

2. RELATED WORKS

In the most recent literature on software estimation, numerous techniques have been proposed to assess the prediction models' accuracy. Researchers have explored various options to solve estimation problems in response to algorithmic model limitations. Some of the existing techniques are given below.

Global software development and applying an amplification cost calculation model based on COCOMO II

[1,2] SNS College of Technology, Coimbatore, India.
[3] PSN College of Engineering and Technology, Tirunelveli, India.
[4] St. Alphonsa college of arts and science, Karinkal, India. Corresponding author
Emails: Sumithra.a.cse@snsct.org, vlakshmi.n.cse@snsct.org, akhilanappathurai@psncet.ac.in, salinda@stalphonsa.edu.in

DOI: 10.59277/RRST-EE.2024.69.4.11

were introduced in [16]. Two datasets evaluate the suggested strategy and yield precise and accurate estimations. Still, the proposed model considers range values instead of exact values.

A method used in software computing to model and calculate a software project's effort was suggested in [17]. A Chinese dataset assesses the proposed approach to increase estimation accuracy. Still, there is an improvement in the recommended method's performance.

In [18], a technique that uses metaheuristics to optimize DNN models for software effort estimation was suggested. Compared to DNN for estimate software, the proposed GWDNNSB model yields better results. To estimate accuracy, GWDNNSB employs nine standard functions. On the other hand, GWDNNSB's convergence rate is high.

An all-encompassing approach to hands-on learning that effectively estimates software development effort was suggested by [19]. When tested on the CHINA and COCOMO81 datasets, the accuracy of the proposed technique is 98% and 93%, respectively. Nevertheless, the suggested approach is very complex.

A semantic web's business effort using COCOMO II, SVM, and NN. SVM and NN are implemented in MATLAB in [20]. The suggested approach yields an effort ratio of 127,729 for the ordinary effort, 10% for the second principle, and 5% for the first principle. However, the suggested approach could be better for high-dimensional data.

[21] proposed a software cost estimation model based on an enhanced self-adaptive differential evolution algorithm with fuzzy C-means. The COCOMO-81 dataset is used to assess the suggested strategy, which improves software development cost prediction. Nevertheless, the proposed approach overfits since it concentrates too much on outliers.

In 2023, [22] suggested a fuzzy neural network emphasizing software project effort estimation. With genetic optimization in elephant farming (GEHO) to estimate the software effort, the suggested NFN approach based on GEHO is developed. Nevertheless, all traits are assumed to be independent in the suggested GEHO-based NFN strategy.

[23] presented approximations derived by precise software efforts that combine hybrid optimization and machine learning methods. It evaluates data from many standard datasets. The AA-SEE approach produces more accurate results. For real-time forecasting, the suggested approach could be more efficient and faster.

[24] suggested an approach that uses software computing to estimate software expenses in agile software development. When tested on the SEERA dataset, the proposed approach received 89.59 % accuracy, 26.8 % profit and loss prediction, 97.06% computational effort, and 97.06% SVM accuracy of 94.45%. The calculation will get more complex if numerous uncertain or uncertain values exist.

[25] suggested an optimization method for software development effort estimation. Nine software datasets] that significantly increase effort estimating accuracy are used to assess the TSoptEE approach. Nevertheless, the computational cost of the suggested TSoptEE approach is significant.

3. THE PROPOSED CORADMO-BASED CETA METHODOLOGY

In this research, a CORADMO-based CETA) was proposed for accurate estimation.

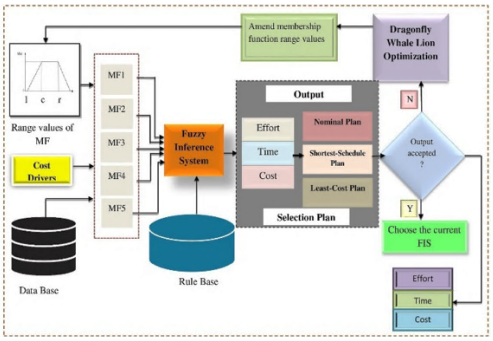


Fig. 2 – Block diagram for CORADMO-based CETA.

The CORADMO-based CETA processes the input and hands over the processed output for optimization. Herein, the software requirement data, cost drivers, constraints, and priorities are given as input to FIS. Then, the processed output is optimized using dragonfly whale lion optimization, which provides the proper estimated effort, time, and cost as an output for software development. Figure 2 depicts the block diagram of the CORADMO-based CETA approach.

3.1 FUZZY INFERENCE SYSTEM

The trapezoidal membership function (MF) and the Mamdani inference technique in this model lead to a fuzzy solution for

$$f(a,b,c,d,\mu)=\begin{cases}0 & \text{when } TV<a \text{ and } TV>d\\ \frac{(a-TV)\mu}{a-b} & \text{when } a\le TV\le b\\ \mu & when\ b\le TV\le c\\ \frac{(d-TV)\mu}{d-c} & \text{when } c\le TV\le d\end{cases} \quad (1)$$

Here, TV represents the user's trust value, a represents the lower bound of the first interval, b represents the upper bound of the first interval and lower bound of the second interval, c represents the upper bound of the second interval and lower bound of the third interval, d represents the upper bound of the third interval, and μ represents a constant multiplier used within each interval. The inputs are divided into valid ranges to construct classes. For example, the requirement data can vary from "minimum" to "maximum".

Table 1

Input parameters with range

Fuzzy Input Parameter	Range		
Requirement Data (RD)	0-0.3 Minimum (min)	0.2-0.65 Average (A)	0.6-1 Maximum (max)
Cost Drivers (CD)	0-0.45 Less (Ls)	0.35-0.65 Average (A)	0.6-1 Heavy (Hv)
Constraints (C)	0-0.5 Low (L)	0.45-0.75 Medium (M)	0.65-1 High (H)
Priorities (P)	0-0.25 Critical (C)	0.3-0.85 Moderate (M)	0.8-1 Severe (S)
Output (O)	0-0.4 Least Cost Plan (LCP)	0.3-0.7 Nominal Plan (NP)	0.6-1 Shortest Schedule Plan (SSC)

The Cost drivers fall between "less" and "heavy," among other ranges. The Constraints can range from "low" to "high". The Priorities can vary from "critical" to "severe". Three fuzzy categories determine trust: shortest schedule plan, nominal plan, and least cost plan. A number between 0 and 1 represents the extent to which a value falls into a particular category (Table 1).

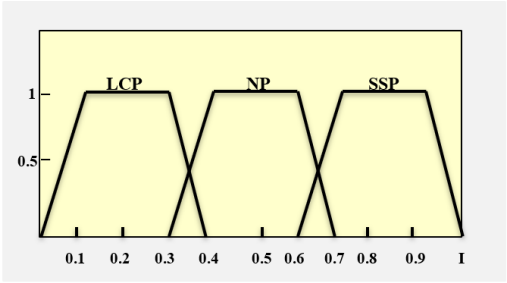


Fig. 3 – Output trust value.

Each fuzzy set has a membership function that establishes the level of MF for a specific value in the fuzzy set. Figure 3 illustrates the output of the trust value, and Fig. 4. depicts the selection plan.

3.2 RULE EVALUATION

Table 2
Sample Fuzzy Rules

1.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
2.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
3.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
4.	IF(RD = = max) & (CD = = Ls) & (C = = H) & (P = = S) → (TV = = SSP)
5.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
6.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
7.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
8.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
9.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
10.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
11.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
12.	IF(RD = = max) & (CD = = Ls) & (C = = H) & (P = = S) → (TV = = SSP)
13.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
14.	IF(RD = = max) & (CD = = Ls) & (C = = H) & (P = = S) → (TV = = SSP)
15.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
16.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
17.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
18.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
19.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
20.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
21.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
22.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
23.	IF(RD = = max) & (CD = = Ls) & (C = = H) & (P = = S) → (TV = = SSP)
24.	IF(RD = = min) & (CD = = H) & (C = = L) & (P = = C) → (TV = = LCP)
25.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)
26.	IF(RD = = max) & (CD = = Ls) & (C = = H) & (P = = S) → (TV = = SSP)
27.	IF(RD = = A) & (CD = = A) & (C = = M) & (P = = M) → (TV = = NP)

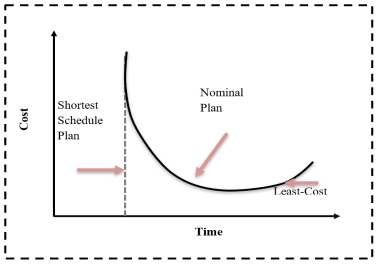


Fig. 4 – Cost estimation graph.

The proposed method evaluates four inputs and three fuzzy sets using the Mamdani system inference engine with trapezoidal MF and a knowledge base containing 27 fuzzy inference rules. Table 2 illustrates a fuzzy rule for determining the degree of trust.

3.3 DRAGONFLY WHALE LION OPTIMIZATION (DWLO) ALGORITHM

DWLO is used to adjust the weights of the neural network's internal parameters to accomplish specific optimization and problem-solving objectives. The DWLO organizational chart is shown in Figure. 5. In this study, the whale optimization method and the dragonfly algorithm are integrated with the lion optimization algorithm to update the optimal solution. The following section provides a mathematical description of the layers of the modified dragonfly whale lion optimization method.

Layers of modified dragonfly whale lion optimization

Layer 1: In this layer, a random input is generated by

$$P = \begin{bmatrix} S_{ig}^{11} & S_{ig}^{12} & \dots & S_{ig}^{1a} \\ S_{ig}^{21} & S_{ig}^{22} & \dots & S_{ig}^{2a} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ S_{ig}^{b1} & S_{ig}^{b1} & \dots & S_{ig}^{ba} \end{bmatrix}, \quad (2)$$

where S_{ig} do the generators produce the total generation.

Layer 2: Equation (3) determines each dragonfly's objective function.

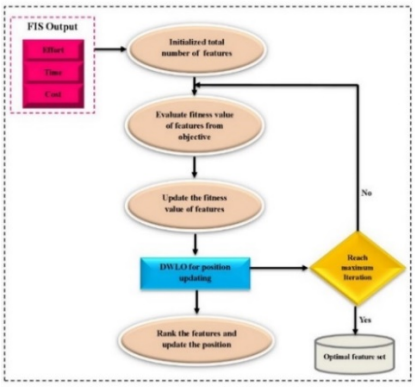
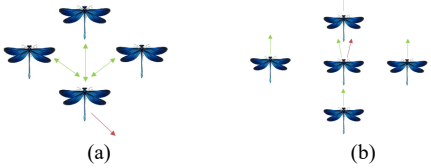


Fig. 5 – Dragonfly whale lion optimization

$$Objective_{Fn} = Min\, f(i,l). \quad (3)$$

The objective of the system is denoted by $f(i,l)$. The best result is attained if the value of the $Objective_{Fn}$ is minimum. The following five factors are used to update the fittest solution.



(a) (b)

Fig. 6 – (a) Separation of dragonfly and (b) alignment of dragonfly.

Layer 3: Using the formula below, find the Sep_i distance between each *i*-th dragonfly.

$$Sep_i = -\sum_{i=1}^{k} P - P_j\,. \quad (4)$$

The preceding equation states that P is regarded as P_j's neighbor if their previous distance is less than their present distance. A neighbor has k neighbors.

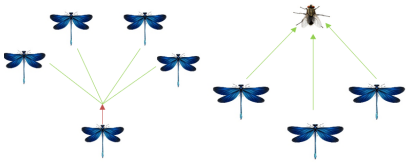
Layer 4: The alignment of $Alig_i$ is verified for each i-th dragonfly individual by applying the subsequent formula.

$$Alig_i = \frac{\sum_{i=1}^{k} V_j}{k}, \quad (5)$$

where, V_j denotes the dragonfly's steady rate of movement.

Layer 5: The formula that follows determines the association Coh_i for every ith dragonfly individual.

$$Coh_i = \frac{\sum_{i=1}^{k} p_j}{k} - p. \quad (6)$$



(a) (b)

Fig. 7 – (a) Cohesion of dragonfly and (b) attraction to food.

Layer 6: The level of attraction of each dragonfly (i) to the food source FS_i is calculated using the following formula:

$$\delta_j = p^{fs} - p\,, \quad (7)$$

$$\sigma_j = p^{ey} - p\,, \quad (8)$$

where p^{ey} represents the enemy source and p^{fs} represents the food source.

Layer 7: In the update process, WOA is encouraged in eq. (9) and (10) to enhance DFA performance accuracy through search behavior

$$H = \left|\vec{X}\overrightarrow{P_{rand}} - \vec{P}\right|, \quad (9)$$

$$P(t+1) = \overrightarrow{P_{rand}} - \vec{Y}\vec{H}, \quad (10)$$

where $\vec{H}$, $\vec{Y}$, and $\vec{X}$ represent the coefficient vectors and $\overrightarrow{P_{rand}}$ indicates the random whales.

LOA utilizes a metaheuristic technique in which a collection of random solutions, known as lions, serves as the foundation for initializing the population. Within this population of N solutions, every solution encompasses α and β features earmarked for enhancement. Further clarification of this concept can be provided as follows:

$$S_L\,(L_n) = [\alpha, \beta], \quad (11)$$

In a population of size N, some lions serve as nomads, while the rest assemble into random Pride groups (P). Within the nomadic lion subset, S% are female, with the remainder being male. As the hunter progressively improves his physical state, the prey (P) attempts to escape and locate a new position, as shown by

$$P' = P + rand(0,1) \times PI \times (P - H). \quad (12)$$

In eq. (6), P' represents the present prey location, *Hunter* (H) signifies the fresh location utilized by the hunter for targeting, and PI symbolizes the percentage of improvement in hunter fitness, as demonstrated in

$$H' = \begin{cases} \text{rand}\big((2 \times P - H), P\big), \\ \quad (2 \times P - H) < P \\ \text{rand}\big(P, (2 \times P - H)\big), \\ \quad (2 \times P - H) > P). \end{cases} \quad (13)$$

Here, H' denotes the current hunter location, while H represents the new hunter location. The improved position of the central hunter is demonstrated by

$$H' = \begin{cases} \text{rand}(H, P), H < P, \\ \text{rand}(P, H), H > P, \end{cases} \quad (14)$$

As each pride has its area, its members preserve the algorithm's exceptional performance in representing the optimal feedback. Equation (15) elucidates the revised position of the female lion (F_L').

$$F_{l,}' = F_{l,} + 2 \times D \times rand(0,1)\,\{R_1\} + U(-1,1) \times \tan(\quad) \times D \times \{R_2\}, \quad (15)$$

$$\{R_1\}.\{R_2\} = 0, \quad (16)$$

$$\|\{R_2\}\| = 1. \quad (17)$$

In this system, *Female Lion* (F_L) symbolizes the lion's current position, whereas D represents the lion's position as determined in the pride territory selection tournament. The value $\{R_1\}$ indicates the lion's initial or former position with relation to $\{R_2\}$. A perpendicular relationship exists between these two vectors, $\{R_1\}$ and $\{R_2\}$. A random number of resident males' mates with each of the specified female lions from the participating pride, c%. Once the genders of individuals have been chosen, two offspring will be generated using equations (18) and (19).

$$OS_j1 = \beta \times F_{L_j} + \sum \frac{1-\beta}{\sum_{i=1}^{NR} S_i} \times M_{L_j}^{i} \times S_i, \quad (18)$$

$$OS_j2 = (1-\beta) \times F_{L_j} + \sum \frac{\beta}{\sum_{i=1}^{NR} S_i} \times M_{L_j}^{i} \times S_i. \quad (19)$$

Here, d stands for size, m for the number of men in the herd, and c for the males. Furthermore, d is set to 0 without a breeding designation and 1 for the i-th male. The standard deviation is 0.1, while the mean is 0.5. M% of genes undergo mutation, where they are substituted with random values. Through this process, LOA generates a population of new cubs inheriting novel features from their parents. Following the defeat, lions with lower fitness are expelled from the pride and become nomads, while those with higher fitness are selected from the population to become resident males. Some random females might become nomads and leave their pride in the migration phase. The best-adapted nomads are reintroduced into the community to replace the killed lion, with freshly converted and existing nomads being graded based on physical state. The mathematical formulation of this concept is

$$x_{i+1} = \begin{cases} \frac{x_t}{a}, if\ x_j < a, \\ \frac{x_t}{(1-a)}, if\ x_i \geq a, \end{cases} \quad i = 1,2, \ldots, D. \quad (20)$$

4. RESULT AND DISCUSSION

This section depicts the results below while applying the proposed CORADMO-based CETA on the NASA 93 dataset. The dataset determines the corresponding MRE, MMRE, and Maximum MRE values.

4.1 DATASET DESCRIPTION

To validate the model, the Promise Repository Nasa93 dataset was chosen. The Nasa93 dataset contains data for 93 projects and 24 features, respectively. The datasets in the Promise Repository have an estimated deviation factor of about 6.6, which indicates that they are not regularly distributed. As a result, the forecasting software development effort is inaccurate.

4.2 PERFORMANCE EVALUATION

A range of metrics are utilized to gauge the efficacy of estimation models. The following outlines three criteria for assessing the performance of the proposed model.

Magnitude of Relative Error

The difference between the actual value of the data set and the expected effort value for a particular project, as established by the proposed framework, is the mean relative error, or MRE. This is ascertained using:

$$MRE = \left|\frac{E_{est}(Pi) - E_{act}(Pi)}{E_{act}(Pi)}\right|, \quad (21)$$

where $E_{est}(Pi)$ is the project's expected work-effort pi and $E_{act}(Pi)$ indicates the actual work effort for project *Pi.*

Mean of Magnitude of Relative Error

To calculate MMRE, apply the formula

$$MMRE = \frac{1}{x}\sum_{i=1}^{n}\frac{E_{est}(Pi)-E_{act}(Pi)}{E_{act}(Pi)}, \quad (22)$$

where $E_{est}(Pi)$ is the project's expected work effort *Pi* and $E_{act}(Pi)$ is the actual project effort, *Pi*, and x is the total number of projects considered.

Prediction level

(Pred(L)) is the percentage of projects with an MRE of less than or equal to L. The calculation applies to

$$Pred(L\%) = \frac{k}{x} * 100. \quad (23)$$

The number of projects in this instance with mean relative error (MRE) equal to or less than L is denoted by k, while the total number of projects is represented by x. L is a threshold number often set to 25, providing a baseline against which to measure the model's performance for evaluating software effort. Generally, the precision of estimation methods is associated with Pred (25%) and inversely related to MMRE.

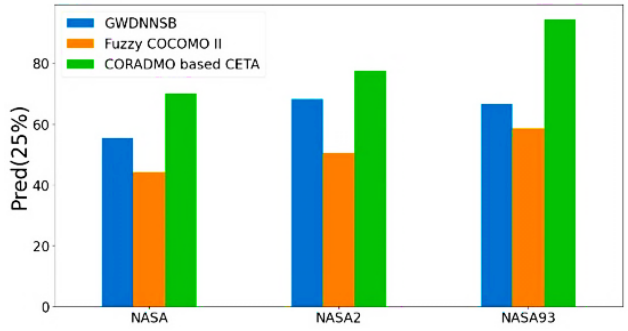


Fig. 8 – Comparison chart for Pred (25 %).

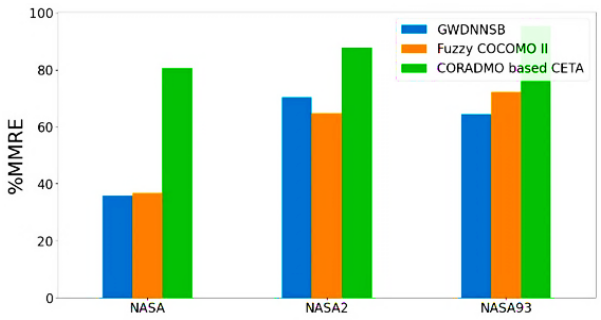


Fig. 9 – Comparison chart for % MMRE.

Figure 8 compares Pred (25 %) values for each of the three models listed in Table 3, while Fig. 9 illustrates the comparison of % MMRE values for the exact models listed in Table 4. These comparisons are conducted using fuzzy logic in COCOMO II % MMRE, further refined by adjusting the fuzzy model's Membership Function parameters via DWLO.

Figure 10 illustrates the Maximum MRE values for each of the three models under consideration, as outlined in Table 5. Optimizing the fuzzy model with DWLO can prove that a more significant proportion of projects with % MRE less than the allowed limit of prediction (25%) may be achieved.

Table 3

Prediction value using three different models and three different datasets.

Pred (25 %)	GWDNNSB	Fuzzy COCOMO II	CORADMO-based CETA
NASA	55.5	44.16	70.17
NASA 2	68.22	50.44	77.48
NASA 93	66.58	58.49	94.52

Table 4

%MMRE value using three different models and three different datasets.

%MMRE	GWDNNSB	Fuzzy COCOMO II	CORADMO based CETA
NASA	35.79	36.8	80.65
NASA 2	70.38	64.85	87.72
NASA 93	64.47	72.22	95.47

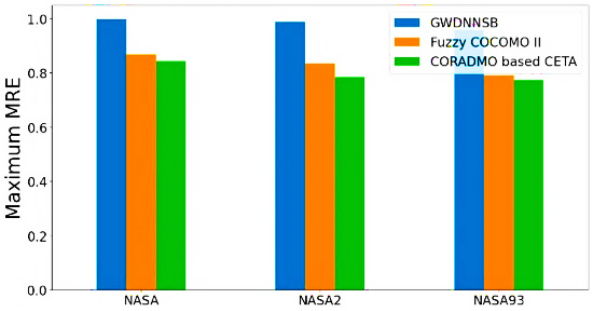


Fig. 10 – Comparison chart for maximum MRE.

Table 5

Maximum MRE value using three different models and three different datasets.

Maximum MRE	GWDNNSB	Fuzzy COCOMO II	CORADMO based CETA
NASA	0.999	0.8664	0.8436
NASA 2	0.9878	0.8349	0.784
NASA 93	0.9572	0.7898	0.7742

The COCOMO II model's forecast accuracy has risen with software calculation techniques like fuzzy logic. The outcomes are significantly enhanced by implementing DWLO to optimize the fuzzy model design. According to the data, Pred (25 %) is maximized, but the MMRE, which represents the total amount of error, is minimized.

5. CONCLUSION

In this paper, the fuzzy logic with the DWLO method is used to handle the uncertainty in defining the input parameters of the COCOMO II model, which leads to the development of a CORADMO-based CETA. To develop the CORADMO-based CETA, the COCOMO II model is fuzzified, and the associated fuzzification parameters are optimized using the DWLO technique. Herein, the software requirement data, cost drivers, constraints, and priorities are given as input to FIS.

Then, the processed output from FIS, such as nominal plan, shortest schedule plan, and least cost plan, are optimized using DWLO, which provides the proper estimated effort, time, and cost as an output for software development. The performance of the CORADMO-based CETA method is measured in terms of Pred (25%), Magnitude of Relative Error, and Mean Magnitude of Relative Error, attaining the values of 80.72%, 87.94%, and 98.13%, respectively. As a result, the model had improved precision, higher accuracy, and increased sensitivity.

Table 1

Notation List

Notation List	**Description**
TV	Trust Value
a	Lower bound of the first interval

b	Upper bound of the first interval
c	Upper bound of the second interval
d	Upper bound of the third interval
μ	Constant multiplier
$f(i,l)$	Objective of the system
$\mathrm{Sep_i}$	Distance between dragonfly
P	Previous distance
$\mathrm{P_j}$	Present distance
$\mathrm{Alig_i}$	Alignment
$\mathrm{V_j}$	Dragonfly's steady rate
$\mathrm{Coh_i}$	Cohesion of dragonfly
k	Neighbor's
$\mathrm{p^{ey}}$	Enemy source
$\mathrm{p^{fs}}$	Food source
$\vec{H}$, $\vec{Y}$, and $\vec{X}$	Coefficient vectors
$\overrightarrow{\mathrm{P_{rand}}}$	Random whales
N	Population size of lion
P	Prey
α and β	Population of solutions
P'	Prey location
H	*Hunter*
PI	Percentage of hunter fitness
H′	Current hunter location
$\mathrm{F'_L}$	Revised position of the female lion
$\mathrm{F_L}$	F*emale* Lion
D	Lion's position
d	Size
m	Number of males in the herd
c	Males
$\mathrm{E_{est}(Pi)}$	Expected work-effort
$\mathrm{E_{act}(Pi)}$	Actual work effort
x	Total number of projects
L	Threshold number

Received on 5 May 2023

REFERENCES

1. M.S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, W. Abdul, *Metaheuristic algorithms in optimizing deep neural network model for software effort estimation*, IEEE Access, **9**, pp. 60309-60327 (2021).
2. W. Rhmann, B. Pandey, G.A. Ansari, *Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms*, Innovations in Systems and Software Engineering, **18**, *2*, pp. 309–319 (2022).
3. S. Hameed, Y. Elsheikh, M. Azzeh, *An optimized case-based software project effort estimation using genetic algorithm.* Information and Software Technology, **153**, pp.107088 (2023).
4. N. Sreekanth, J. Rama Devi, K.A. Shukla, D.K. Mohanty, A. Srinivas, G.N. Rao, A. Alam, A. Gupta, *Evaluation of estimation in software development using deep learning-modified neural network*, Applied Nanoscience, **13**, *3*, pp. 2405–2417 (2023).
5. S. Kassaymeh, M. Alweshah, M.A. Al-Betar, A.I. Hammouri, M.A. *Al-Ma'aitah, Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques*, Cluster Computing, **27**, *1*, pp. 737–760 (2024).
6. M. Hassanali, M. Soltanaghaei, T. Javdani Gandomani, F. Zamani Boroujeni, *Software development effort estimation using boosting algorithms and automatic tuning of hyperparameters with Optuna.* Journal of Software: Evolution and Process, pp. e2665.
7. M. Zorzetti, I. Signoretti, L. Salerno, S. Marczak, R. Bastos, *Improving agile software development using user-centered design and lean startup*, Information and Software Technology, **141**, pp.106718 (2022).
8. A. Yasmin, *Cost adjustment for software crowdsourcing tasks using ensemble effort estimation and topic modelling*, Arabian Journal for Science and Engineering, pp. 1–36 (2024).
9. A. Jaiswal, J. Raikwal, P. Raikwal, A *hybrid cost estimation method for planning software projects using fuzzy logic and machine learning*, International Journal of Intelligent Systems and Applications in Engineering, **12**, *1*, pp. 696–707 (2024).
10. E. Venson, B. Clark, B. Boehm, *The effects of required security on software development effort*, Journal of Systems and Software, **207**, pp.111874 (2024).
11. C.H. Anitha, N. Parveen, *Deep artificial neural network based multilayer gated recurrent model for effective prediction of software development effort*, Multimedia Tools and Applications, pp. 1–27, (2024).
12. M. Hassanali, M. Soltanaghaei, T.J. Gandomani, F.Z. Boroujeni, *Software development effort estimation using boosting algorithms and automatic tuning of hyperparameters with Optuna*, Journal of Software: Evolution and Process, p.e2665.
13. N. Zidane, S.L. Belaid, *A new fuzzy logic solution for energy management of hybrid photovoltaic/battery/hydrogen system*, Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **67**, *1*, pp. 21–26 (2022).
14. M. Jesi, A. Appathurai, M. Kumaran, A. Kumar, *Load balancing in cloud computing via mayfly optimization algorithm*, Revue Rev. Roum. Sci. Techn. – Électrotechn. Et Énerg., **69**, *1*, pp. 79–84 (2024).
15. K. Upreti, U.K. Singh, R. Jain, K. Kaur, A.K. Shar-ma, *Fuzzy logic-based support vector regression (SVR) model for software cost estimation using machine learning*, ICT Systems and Sustainability: Proceedings of ICT4SD 2021, **1**, pp. 917–927 (2022).
16. J.A. Khan, S.U.R. Khan, T.A. Khan, I.U.R. Khan, *An amplified COCOMO-II based cost estimation model in global software development context*, IEEE Access, **9**, pp. 88602–88620 (2021).
17. S. Sharma, S. Vijayvargiya, *Applying soft computing techniques for software project effort estimation modelling*, Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS 2019, pp. 211–227 (2021).
18. M.S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, W. Abdul, *Metaheuristic algorithms in optimizing deep neural network model for software effort* estimation, IEEE Access, **9**, pp. 60309–60327 (2021).
19. P. Suresh Kumar, H.S. Behera, J. Nayak, B. Naik, *A pragmatic ensemble learning approach for effective software effort estimation*, Innovations in Systems and Software Engineering, **18**, *2*, pp. 283–299 (2022).
20. N. Malik, S.K. Goyal, V. Malik, *Semantic web undertaking effort estimation utilizing COCOMO II, SVM and NN*, Cyber Technologies and Emerging Sciences: ICCTES **2021**, pp. 351–361 (2022).
21. S.K. Gouda, A.K. Mehta, *Software cost estimation model based on fuzzy C-means and improved self-adaptive differential evolution algorithm*, International Journal of Information Technology, **14**, *4*, pp. 2171–2182 (2022).
22. S. Sharma, S. Vijayvargiya, *An optimized neuro-fuzzy network for software project effort estimation*, IETE Journal of Research, **69**, *10*, pp. 6855–6866 (2023).
23. K.H. Kumar, K. Srinivas, *An accurate analogy based software effort estimation using hybrid optimization and machine learning techniques*, Multimedia Tools and Applications, **82**, *20*, pp. 30463–30490 (2023).
24. S. Kumar, M.P. Singh, *An improved technique for software cost estimations in agile software development using soft computing* (2023).
25. P.J. Shermila, A. Ahilan, M. Shunmugathammal, J. Marimuthu, *DEEPFIC: food item classification with calorie calculation using dragonfly deep learning network*, Signal, Image and Video Processing, **17**, *7*, pp. 3731–3739 (2023).
26. A.S. Ghazanfar, X. Cheng, *Brain aneurysm classification via whale optimized dense neural network,* International Journal of Data Science and Artificial Intelligence, **2**, *2*, pp. 63–67 (2024).
27. K.B. Shah, S. Visalakshi, R. Panigrahi, *Seven class solid waste management-hybrid features based deep neural network*, International Journal of System Design and Computing, **1**, *1*, pp. 1–10 (2023).